# Búsqueda local

## Documentación Práctica 1

Inteligencia Artificial


**<< TRUCKS ORDERS >>**

**Alumnos:**
Miquel Perelló Nieto
Oscar Rodríguez Pestaña

**Fecha:**
04 - 04 - 2011

# Índice

# 1. Descripción del problema

Una compañía dispone de un almacén central del que diariamente se prepara un reparto a seis centros de producción donde necesitan distintas cantidades de material (fig. 1). Nosotros como compañía de transporte somos subcontratados para hacer éste transporte diario y por lo tanto necesitamos sacar el máximo beneficio de las entregas, intentando en todo momento entregar el máximo de paquetes antes de la hora marcada por los centros demandantes.

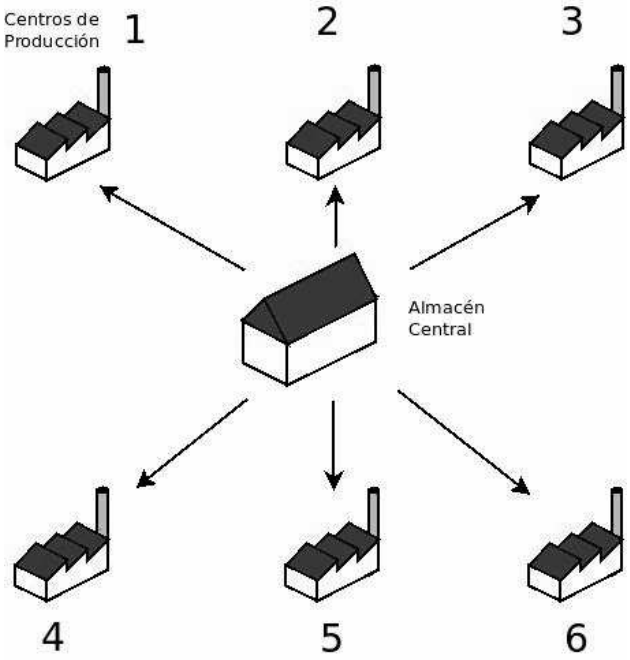



*Figura 1: Esquema general Empresa contratadora*

Cada uno de los centros prepara un pedido diario que nos es entregado, estas peticiones cuentan con la hora limite de entrega y la cantidad ( en kilogramos ) del material requerido. El tamaño de los paquetes demandados siempre será con saltos discretos de 100kg con un máximo de 500kg de peso. El precio del transporte de los distintos paquetes incrementará con el peso de éste según la tabla siguiente (tabl. 1):

| Peso | Precio |
|---|---|
| 100 y 200 kg | 1 x peso euros |
| 300 y 400 kg | 1,5 x peso euros |
| 500 kg | 2 x peso euros |

*Tabla 1: Precios según peso*

Nuestra compañía de transporte para realizar todos los pedidos cuenta con una flota de 60 camiones de distintos tamaños o capacidades para asignar a este cliente. Las capacidades de nuestros camiones son de 500, 1000 y 2000 Kg (fig. 2) , pero aunque el total de camiones asignables es de 60, la cantidad individual de cada tipo puede variar de un día a otro. Cada uno de los camiones tendrá asignado uno de los 6 centros de producción y una de las 10 horas de entrega, por lo que nunca podrán coincidir dos camiones en un mismo horario y centro.

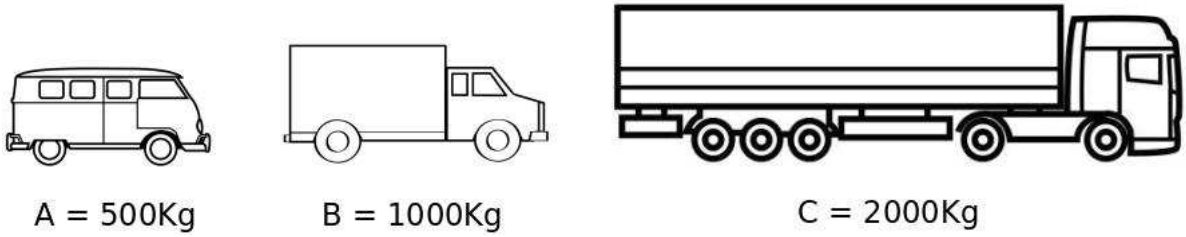



*Figura 2: Tipos de camiones disponibles*

Los horarios de entrega estarán comprendidos entre las 8 de la mañana y las 5 de la tarde, nuestro objetivo será asignar la hora a la que se realizará el envío al centro de producción indicado. En el caso de entregar un paquete más tarde de la hora límite de entrega, se nos sancionará con un 20% del precio del paquete por hora de retraso, y si no se puede realizar la entrega en el mismo día además de la sanción (aplicada hasta las 5 de la tarde) se tendrá que descontar también el precio total de dicho pedido.

A continuación se puede observar un ejemplo ( resumido ) de las peticiones diarias por los distintos centros de producción (tabl. 2) Ejemplo de petición diaria:

| **Centro de Producción** | **Peso** | **Hora Límite** | **Factor** | **Precio** |
|---|---|---|---|---|
| [ 1 – 6 ] | [ 100 – 500 ] kg | [ 08:00 – 17:00 ] | (x1; x1,5 ;x2) | euros |
| 1 | 400 | 08:00 | 1,5 | 600 |
| 1 | 200 | 10:00 | 1 | 200 |
| 1 | 500 | 11:00 | 2 | 1000 |
| 2 | 400 | 08:00 | 1,5 | 600 |
| 2 | 400 | 08:00 | 1,5 | 600 |
| … | … | … | … | … |
| 6 | 300 | 13:00 | 1,5 | 450 |
| 6 | 500 | 17:00 | 2 | 1000 |

*Tabla 2: Tipos de camiones disponibles*

Ahora un nuevo ejemplo que muestra como afecta un retraso en una petición (tabl. 3). Dada una petición de 100kg de peso y con hora límite de entrega a las 10:00 de la mañana, veamos como influye en la ganancia la hora en la que se realiza la entrega. Ejemplo de retraso de petición:

| Entrega | 08:00 | 09:00 | **10:00** | 11:00 | 12:00 | 13:00 | 14:00 | 15:00 | 16:00 | 17:00 | No Entregada |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ganancia | 100 | 100 | **100** | 80 | 60 | 40 | 20 | 0 | -20 | -40 | -140 |

*Tabla 3: Efectos de retraso en una entrega*

# 2. Representación del estado

## 2.1. Elementos:

Para poder representar el problema deberemos disponer de diversos elementos, intentando siempre minimizar los costes de espacio ya sea por el aprovechamiento de datos estáticos o por la simplificación de variables redundantes o replicadas ya que de ello dependerá la eficiencia de nuestro programa tanto en tiempo como en espacio. Los elementos elegidos para representar el estado son los siguientes:

**Una lista de pedidos**:

Debido a que toda la información que recibiremos sobre los pedidos no variará, ésta lista será **estática** y vendrá determinada por la entrada, aquí tendremos toda la información necesaria sobre cada uno de los pedidos, podremos evaluar entonces la urgencia de los paquetes, sus destinos y su peso.

Información que contendrá:

- Almacén demandante.
- Hora límite de entrega.
- Peso del material.

El **espacio ocupado** por esta lista es de:

número de pedidos * ( almacén + hora + peso )

*Coste: Θ ( #pedidos )*

**Un horario de H x C**:

Todo el problema se basa en organizar una plantilla de horario donde se indicara la organización de pedidos y camiones que se ha decidido, esto se plasmará en ésta matriz. Éste será un horario guardado de forma **dinámica**, pues se irá modificando en cada estado generado. Mediante él podremos controlar el tipo de camiones asignados a cada una de las posiciones, así como los pedidos que contienen esos camiones. Gracias a éste se podrán intercambiar el tipo de camiones entre distintos almacenes u horarios.

Información que contendrá:

- H son las 10 horas disponibles de 8:00 a 17:00
- C los distintos centros de producción de los 6 existentes.

Cada una de las posiciones de este horario contienen:

- Tipo de camión asignado.
- Una lista de pedidos asignados en ésta posición.

Por lo tanto tenemos que el **espacio ocupado** por este horario es de :

H x C x camión + numero de pedidos asignados.

60 x camión + 1.200

*El número de pedidos asignados puede ser acotado teniendo en cuenta que solo puede haber 60 camiones (en el peor de los casos de 2.000 kg), y todos los pedidos asignados del menor de los pesos (100 kg), de modo que la cota superior sería de:

60 * 2.000 / 100 = 60 * 20 = 1200

por lo tanto, el número de pedidos asignados será menor o igual a 1200.

*Coste: Θ ( #centros * #horas + #pedidos_asignados)*

**Una lista con la posición de los pedidos** en el horario:

Necesitamos saber si un pedido a sido o no asignado y en caso afirmativo conocer en que hora y centro de producción se encuentra, ésta doble información se consigue utilizando un número que indique su posición si ya ha sido asignado y un número fuera del dominio para determinar que aún no lo ha sido. Ésta lista será **dinámica**, cada uno de los estados generados tendrá su lista que nos indicará en que situación se encuentran todos los pedidos.

Información que contendrá:

- Posición del pedido en el horario [ -1..59 ]

*La posición -1 indica que no ha sido asignado, cualquier otro valor indica la posición dentro del horario desde la 0 a las 8:00 en el centro 1, hasta la 59 a las 17:00 en el centro 6.

El **espacio ocupado** por esta lista es de :

número de pedidos * ( posición )

*Coste: Θ ( #pedidos )*

**Elementos para la obtención de los resultados del problema:**

Será necesario mantener en cada estado diversas variables que contengan los resultados del análisis que realiza el programa. Como éstas variables contienen un resultado parcial de cada estado generado, las guardaremos de forma **dinámica**. Es importante entonces que el coste en espacio sea mínimo para que no represente un problema al generar los distintos estados del problema.

Las variables utilizadas para éste propósito son: el número de pedidos asignados, que llevará el conteo del total de pedidos que ya se han asignado a alguna hora y centro concretos; los beneficios, que llevarán las ganancias o perdidas acumuladas del estado concreto; y la diferencia de horas, encargada de mantener el valor absoluto de la diferencia entre las horas límites de entrega de las peticiones y las horas en las que se ha realizado. En el caso de que no se realizara la entrega se contarían las horas como si la entrega se realizase a las 8:00 del día siguiente

Información que contendrá:

- Número de pedidos asignados.
- Beneficios.
- Diferencia de horas.

El **espacio ocupado** de todos estos elementos es de:

1 + 1 + 1

*Coste total: Θ ( 1 )*

**Elementos secundarios**:

Además de los elementos ya comentados, dispondremos de variables extra para ayudar con el análisis del problema que no afectan a su coste asintótico. Como los valores de éstas variables serán comunes a todos los estados del problema también se guardarán de forma estática.

Dichas variables son: una semilla, que será utilizada para fijar los estados aleatorios para los experimentos; contador de sucesores, para poder consultar el número de sucesores que ha generado el problema; el número de pedidos, que contiene la cantidad total de pedidos que nos han solicitado; los tipos de camiones, que a su vez contienen información de la capacidad total de la que disponen y la cantidad de ese tipo de camión que conforman la flota; y una lista de los beneficios potenciales de cada centro, con el valor acumulado de todos los pedidos del centro en cuestión en caso de que todos fueran entregados a tiempo.

Información que contendrá:

- Semilla.
- Contador de sucesores.
- Número de pedidos.
- Tipos de camiones. (tipo + capacidad + cantidad)
- Beneficios potenciales de cada centro.

El **espacio ocupado** de todos estos elementos es de:

1 + 1 + 1 + (3 * tipos de camiones) + numero de centros

*Coste total: Θ ( #tipos_de_camiones + #centros )*

# 2.2. Análisis del tamaño del espacio de búsqueda:

Con los elementos arriba comentados conseguimos que el espacio ocupado en memoria junto con la eficiencia de cálculo sean bastante óptimos. Debemos diferenciar los costes en espacio dinámicos y estáticos ya que los que crecerán al hacer la búsqueda replicándose en nuevos estados generados serán los dinámicos, mientras que los estáticos permanecerán fijos durante todo el proceso de análisis.

**Costes espaciales estáticos:**

N * (alm + hora + peso) + seed + suc + pedidos_s + T * (tipo_c + cap_c + cant_c) + C =
N * (1*1B + 1*1B + 1*2B) + 1*4B + 1*4B + 1*2B + T * (1*1B + 1*2B + 1*2B) =
N * ( 4B ) + 4B + 4B + 2B + 3 * ( 5B ) =
4N bytes + 10 bytes + 15 bytes =
4N bytes + 25 bytes

**Costes espaciales dinámicos:**

H * C * ( tipo_c + (Na * pedido) ) + N + pedidos_a + benef + dif_hora =
H * C * ( 1*1B + (Na * 1*4B) ) + N bytes + 1*2B + 1*4B + 1*4B =
10 * 6 * ( 1B + 4Na bytes) + N + 10B =
60B + 240Na bytes + N bytes + 10B =
N bytes + 240Na bytes + 70 bytes

Teniendo en cuenta que la cota superior de Na es 1200 (como ya calculamos anteriormente) podemos estimar un coste de:

N bytes + 240Na bytes + 70 bytes =
N bytes + (240 * 1200) bytes + 70 bytes =
N bytes + 288000 bytes + 70 bytes =
N bytes + 288070 bytes =
N bytes + ~280 kB

| | | |
|---|---|---|
| *N:* | *número total de pedidos.* | |
| *T:* | *número de tipos de camión.* | |
| *H:* | *número de horas del horario.* | |
| *C:* | *número de centros de producción.* | |
| *Na:* | *número de pedidos asignados.* | |
| *alm:* | *almacena el id del almacén que realiza el pedido.* | *[byte = 1 bytes]* |
| *hora:* | *almacena la hora en que se debe realizar el pedido.* | *[byte = 1 byte]* |
| *peso:* | *almacena el peso del pedido.* | *[short = 2 bytes]* |
| *seed:* | *almacena el número para generar la aleatoriedad.* | *[entero = 4 bytes]* |
| *suc:* | *almacena la cantidad de estados que se generan.* | *[entero = 4 bytes]* |
| *pedido:* | *puntero a un pedido asignado.* | *[entero = 4 bytes]* |
| *pedidos_s:* | *almacena la cantidad total de pedidos solicitados.* | *[short = 2 bytes]* |
| *pedidos_a:* | *almacena la cantidad total de pedidos asignados.* | *[short = 2 bytes]* |
| *posición :* | *almacena la posición en la matriz.* | *[byte = 1 byte]* |
| *tipo_c:* | *almacena el id del tipo de camión.* | *[byte = 1 byte]* |
| *cap_c:* | *almacena la capacidad máxima de un camión.* | *[short = 2 bytes]* |
| *cant_c:* | *almacena la cantidad total de camiones de un tipo.* | *[short = 2 bytes]* |
| *benef:* | *almacena las ganancias o perdidas acumuladas.* | *[entero = 4 bytes]* |
| *dif_hora:* | *almacena la diferencia entre horas límite y de entrega* | *[entero = 4 bytes]* |

*NOTA: Para mayor eficiencia del espacio reducimos en lo posible el uso de enteros por shorts o bytes siempre que podamos asegurar que no afecte al proceso del programa.*

**Estados:**

El número de estados posibles es difícil de indicar por la complejidad del problema, pero podemos encontrar cotas superiores e intentar mejorarlas mediante algunas restricciones.

Primero tenemos que el número de peticiones que se pueden tener asignadas en cada estado tiene una cota superior que es en el momento en que todos los camiones son de capacidad 2.000kg y todos los pedidos asignados son de 100kg, lo que nos ofrece la cota de 1.200 peticiones.

Con ésta cota podemos decir que las N peticiones totales del problema se pueden asignar como mucho cogiendo todas las combinaciones de 1.200 de ellas que se representa como N sobre 1.200:

$$\binom{N}{1200}$$

Cada una de estas combinaciones podemos partirla en 10 horas y 6 centros por lo tanto podemos partirla por 60 sitios distintos, a este tipo de particiones se les llama Número de Stirling de Segunda especie y se representa con S(n,k), se definen como la cantidad de maneras que existen de hacer una partición de un conjunto de n elementos en k subconjuntos. En nuestro caso n cogerá el valor de 1.200 y k son las 60 particiones. Este es un valor constante, llamando a este valor A nos queda:

$$O\left(\binom{N}{1200} * A\right) = O\left(\binom{N}{1200}\right) = \left(\frac{n!}{(1200! * (n-1200)!)}\right) = \mathbf{O(n!)}$$

## 2.3. Justificación:

Podemos suponer que varios de los elementos del estado va a ser inferior al máximo número representable con un entero de 4 bytes para 2.147.483.648 valores (positivos y otros tantos negativos) por lo que se hace uso de variables short de 2 bytes con 32.768 valores, o de variables byte de un único byte con 128 valores. Entre ellos están valores fijos que se sabe que no variarán o valores que no deberían superar un cierto límite.

Toda la información que nos interesa se encuentra en un horario en forma de matriz de H x C donde cada casilla corresponde a una hora y un centro concretos, dentro de dicha casilla se encuentra la información de que tipo de camión es el que se encarga de hacer la entrega y una lista de los pedidos que habrá que entregar. El camión se identifica con un byte que determina que tipo de camión se trata sin tener más información redundante como su peso que ya podremos averiguar por su tipo. La lista de pedidos son punteros a datos estáticos porque los datos concretos sobre cada pedido no varían y no hay ninguna necesidad de replicar dicha información.

Por otro lado tenemos la lista que determina que pedidos han sido ya asignados y cuáles quedan aún por colocar. Podríamos llegar a tener problemas si el número de pedidos totales fuera muy grande, ya que realmente como mucho habría 1.200 de ellos asignados, y el resto en nuestra representación estaría malgastando espacio por cada uno de los estados generados. Considerando que en los casos reales no se pedirían una cantidad 2 órdenes superior al número de pedidos capaces de ser entregados hemos creído conveniente primar la velocidad en la búsqueda del pedido de forma constante al espacio que puedan malgastar pedidos no asignados. Para ello se mantiene la información de los pedidos de forma estática mientras que lo único que varía dinámicamente son las posiciones en donde se encuentra cada uno de ellos, de esta forma también se reduce un coste constante para encontrar un pedido por un acceso directo. Para optimizar aún más evitando el uso de dos variables para determinar las coordenadas en la matriz, se ha utilizado un solo valor que indica la "casilla" en la que se encuentra, numeradas del 0 al 59.

De todos modos, en el caso de usar alguna estructura para mantener los pedidos asignados/desasignados, existiría una alternativa plausible en caso de que el número de pedidos fuera de tal magnitud, y seria cambiando la lista de pedidos con su posición por un Árbol B+ solo con los pedidos asignados. De este modo solo se gastaría el espacio de pedidos asignados, teniendo una cota superior de 1.200, pero comprobar que un pedido estuviera asignado, asignarlo y/o desasignarlo conllevaría un coste log 1.200 = ~ 10

Con la estructura propuesta disponemos de toda la información invariable de forma estática para consultarla desde cualquier estado generado sin necesidad de tener dichos datos duplicados, y la información dinámica de la que dispone cada uno de los estados se ha procurado reducir lo máximo posible con ésta representación y con el aprovechamiento de los diferentes tipos de variables.

# 3. Representación de los Operadores

Para poder generar todos los estados necesarios para encontrar una solución adecuada u óptima nos valdremos de los siguientes operadores:

**- ColocarPetición:**

Mediante éste operador podremos asignar una petición pendiente a una posición del horario (centro y hora); siempre y cuando se cumplan ciertas restricciones:

- La petición aún no ha sido asignada.
- La petición cabe en el camión asignado en esa posición del horario.
- La petición se debe entregar en el almacén indicado.

**- QuitarPetición:**

Mediante éste operador podremos eliminar la asignación que tenía en el horario. Para poder recorrer todo el espacio de estados siempre deberá ir acompañado de otro de los operadores ya que tan solo quitar hace que empeore la situación actual pero permite una mejora potencial para otro operador. Éste operador tan solo tiene una restricción:

- La petición está asignada a alguna posición del horario.

**- IntercambiarCamiones:**

Mediante éste operador permitiremos explorar todos los espacios posibles, permitiremos que el camión origen y el destino sean de distintos almacenes y distintas horas pudiendo de este modo explorar todos los posibles estados. Este operador lo aplicaremos si cumple las siguientes restricciones:

- Los tipos de los camiones a intercambiar serán distintos.
- Los pedidos de los camiones siguen cabiendo al hacer el cambio.

Con los tres operadores anteriores podemos deducir el factor de ramificación de nuestro espacio de estados, éste factor indica todos los posibles estados que se pueden generar a partir de un estado concreto y con los operadores comentados:

F.R. = ColocarPetición + QuitarPetición + IntercambiarCamiones =
|P|*H + |P| + (H*C)*(H*C - 1)/2 =
10|P| + |P| + 60*59/2 =
11|P| + 60*59/2 =
11|P| + 1770 estados

*|P|:* *número de peticiones*
*H:* *número de horas*
*C:* *número de centros*

Como veremos a continuación éste cálculo no es del todo correcto por dos razones principales, una es que las restricciones de cada operador filtran en gran medida la generación de dichos estados por lo que la cifra dada se podría considerar como una cota superior. La otra razón es que por la naturaleza de los heurísticos utilizados el operador de quitar una petición siempre genera un estado con un valor heurístico peor, por lo que jamás será considerado como un sucesor. Algo similar pasa con el intercambio de camiones, el hecho de cambiar un camión por otro para el heurístico no supone ninguna diferencia y al obtener el mismo valor le sucederá lo mismo que al operador de quitar petición y tampoco se le considerará como un sucesor potencial.

Para solventar éste problema se utilizan conjuntos de operadores en los que, antes de guardar un sucesor, se utilizan varios operadores juntos, a continuación se muestran los dos conjuntos seleccionados. El primero de ellos es mucho más costoso porque recorre más profundamente el espacio de estados, mientras que el segundo sacrifica parte del espacio de estados para ganar en velocidad reduciendo significativamente su ramificación y no generando tantos estados para analizar en cada iteración.

## 3.1. Conjunto 1:

**- Añadir pedido en una posición del horario.**

Mediante este operador podremos asignar una petición pendiente a una posición del horario; siempre y cuando se cumplan ciertas restricciones:

- La petición aún no ha sido asignada.
- La petición cabe en el camión asignado en esa posición del horario.
- La petición se debe entregar en el almacén indicado.

Ramificación: |P|*H

**- Quitar pedido en una posición del horario y ponerla en otra hora.**

Con este operador permite explorar el espacio de soluciones en estados que contienen espacios vacíos. Éste operador tiene las siguientes restricciones:

- La petición está asignada a alguna posición del horario.
- La hora de destino es distinta a la hora de la posición anterior.
- El almacén del horario es el mismo que el de la petición.
- La petición cabe en el camión asignado.

Ramificación: |P|*(H-1)

**- Intercambiar las horas de entrega de dos pedidos del mismo centro.**

Este operador nos permite explorar espacios de estado totalmente llenos, o semiocupados.
Las restricciones de este operador son:

- Los dos pedidos son del mismo centro.
- Están asignados en camiones de distintas horas.
- Las capacidades de los camiones contenedores y su peso permiten hacer el cambio.

Ramificación: |P|*(|P|-1)

**- Sustituir un pedido asignado por uno que no lo estaba.**

Éste operador nos permite seguir explorando estados incluso teniendo todos los camiones llenos. Las restricciones de este operador son:

- Una petición aún no ha sido asignada, la otra sí.
- Los dos pedidos pertenecen al mismo centro.
- El pedido ha asignar cabe en el camión.

Ramificación: (|P|-1)*(|P|-1)

**- Intercambiar camiones de distinto tamaño y asignar un pedido al mayor de ellos.**

Mediante este operador podemos movernos en espacios de estado semicompletos.
Sus restricciones son:

- La capacidad de los camiones permite hacer el cambio.
- Los camiones son de distinto tamaño.
- Existe un pedido no asignado que puede introducir-se en uno de los camiones.

Ramificación: (60*59/2)*|P|

|*P*|: *número de peticiones*
*H:* *número de horas*

**Ramificación del conjunto 1:** |P|*H + |P|*(H-1) + |P|*(|P|-1) + (|P|-1)*(|P|-1) + (60*59/2)*|P|

Para facilitar el cálculo se puede hacer una aproximación donde (|P|-1) < |P| y de esta forma podemos determinar que la ramificación será como máximo:

$$10|P| + 9|P| + |P|^2 + |P|^2 + 1770|P| = \mathbf{2|P|^2 + 1789|P|}$$

## 3.2. Conjunto 2:

**- Añadir pedido en una posición del horario.**

Mediante este operador podremos asignar una petición pendiente a una posición del horario; siempre y cuando se cumplan ciertas restricciones:

- La petición aún no ha sido asignada.
- La petición cabe en el camión asignado en esa posición del horario.
- La petición se debe entregar en el almacén indicado.

Ramificación: |P|*H

**- Quitar pedido en una posición del horario y ponerla en otra hora.**

Con este operador permite explorar el espacio de soluciones en estados que contienen espacios vacíos. Éste operador tiene las siguientes restricciones:

- La petición está asignada a alguna posición del horario.
- La hora de destino es distinta a la hora de la posición anterior.
- El almacén del horario es el mismo que el de la petición.
- La petición cabe en el camión asignado.

Ramificación: |P|*(H-1)

**- Intercambiar camiones de distinto tamaño y asignar un pedido al mayor de ellos.**

Mediante este operador podemos movernos en espacios de estado semicompletos.
Sus restricciones son:

- La capacidad de los camiones permite hacer el cambio.
- Los camiones son de distinto tamaño.
- Existe un pedido no asignado que puede introducir-se en uno de los camiones.

Ramificación: (60*59/2)*|P|

|*P*|: *número de peticiones*
*H:* *número de horas*

**Ramificación del conjunto 2:** |P|*H + |P|*(H-1) + (60*59/2)*|P|

Para facilitar el cálculo se puede hacer una aproximación donde (|P|-1) < |P| y de esta forma podemos determinar que la ramificación será como máximo:

$$10|P| + 9|P| + 1770|P| = \mathbf{1789|P|}$$

# 4. Funciones Heurísticas

Como explicamos en la representación de los operadores, a causa de éstos heurísticos los operadores de quitar petición y intercambiar camiones no funcionan por si solos, esto provoca que ambos operadores deban ir seguidos de un añadir petición que si incremente el valor de las ganancias.

El hecho de que quitar una petición del horario no mejore ninguno de los heurísticos provoca que dado un estado, el número de peticiones asignadas nunca pueda disminuir. Para solventar esto se podría valorar el hecho de quitar la petición como una posible mejora (deja espacio libre para otras combinaciones con mejores resultados), haciendo que el heurístico mejore en función de la cantidad de espacios libres disponibles, pero esto conlleva riesgos. El riesgo de intentar valorar el quitar una petición puede llevar a que el heurístico considere que sacar peticiones del horario es mejor que ponerlas llegando así a una solución semi-vacía que él la consideraría como la mejor solución.

## 4.1. Maximizar ganancia

Éste heurístico se encarga de determinar que estado de todos los generados es el que puede llevar a una mayor ganancia para convertirlo en el siguiente sucesor. Para evaluar cuál de ellos es el ideal consulta las ganancias acumuladas de cada estado. Estas ganancias se han ido modificando con cada uso de los operadores, tanto al añadir como al quitar un pedido del horario se ha modificado su valor acumulado.

El cálculo de ganancias se realiza primeramente considerando cada una de las peticiones como no entregada generando así la mínima ganancia posible (que en el caso inicial es negativa); y con cada una de las peticiones que se va asignando se incrementa el valor de ganancias que produce junto con las perdidas que estaba generando antes de ser asignada. Por otro lado al quitar una petición del horario se hace el proceso inverso, se decrementa el valor de la ganancia que generaba por estar asignada junto con las perdidas que vuelve a producir por no estar en el horario. Todos estos cálculos tienen en cuenta tanto la tabla de “precio por peso”, como la fórmula de “penalización por retraso”.

Debido a las propiedades de la estructura AIMA el signo del heurístico ha sido cambiado para que maximice en lugar de minimizar.

**Fórmula heurística:** -(ganancias)

## 4.2. Minimizar diferencia horaria

Éste heurístico se encarga de determinar que estado de todos los generados hace descender la diferencia que hay entre la hora límite de entrega y la hora en la que se realiza dicha entrega, convirtiéndolo así en el siguiente sucesor. Para evaluar cuál de ellos es el ideal consulta la diferencia de horas absolutas de cada estado. Éste valor se han ido modificando con cada uso de los operadores, tanto al añadir como al quitar un pedido del horario.

El cálculo de diferencia horaria se realiza considerando en un principio a cada una de las peticiones como no entregada generando así la máxima diferencia horaria posible; y con cada una de las peticiones que se va asignando ésta diferencia se reduce. Por otro lado al quitar una petición del horario se hace el proceso inverso, eliminando la mejora que había producido e incrementando la diferencia que ahora se produce por el hecho de no entregarla. Todos estos cálculos tienen en cuenta el hecho de que no entregar una petición se considera como entregada a las 8:00 del día siguiente.

**Fórmula heurística:** +(horas_absolutas)

# 5. Elección y Generación del Estado Inicial

Es bien sabido que iniciar la búsqueda partiendo de un estado cercano al óptimo puede acelerar mucho la búsqueda de la solución, aunque también es cierto que podemos alejar-nos de la solución si en problemas muy difíciles no hemos tenido algún punto en mente.

Vamos a especificar dos generadores básicos que para ciertas ocasiones donde no se puede imaginar un espacio de soluciones real, no queda más remedio que aplicar-los. No es éste el caso, en el que el problema se puede orientar fácilmente a algún espacio de soluciones bueno.

A continuación vamos a exponer los distintos modos de inicio pero antes, para poder calcular los costes de todos los generadores, vamos a resumir algunos de los costes, de este modo en la explicación individual de cada generador nos ahorraremos repetir los procesos de cálculo:

**O(capacidad_total_camiones) = O(1200) = O(1);**

Ésta es una cota superior del número de pedidos capaces de albergar entre todos los camiones. Ésta cifra se deriva de lo siguiente: En el peor de los casos todos los camiones tendrán la capacidad máxima de 2.000kg, y los pedidos menos pesados son de 100kg, por lo tanto nos queda que como mucho cada camión podrá transportar 2.000 / 100 pedidos = 20 pedidos / camión. También tenemos que en todos los problemas existen sólo 60 camiones lo que hace que el número total de pedidos asignados en los camiones sea siempre igual o inferior a: 60 * 20 = 1.200 pedidos.

**Θ(#pedidos) = Θ(n);**

Como el número de pedidos es variable no podemos darle una cota superior, por lo tanto tiene que ser Θ(n), donde n es el numero de pedidos ( tal vez llamar-le Θ(p) ).

## 5.1. Empty

Éste algoritmo de inicio es como su nombre indica dejar el espacio inicial vacío para que se vaya llenando durante la ejecución. Como decíamos antes puede ser en algún caso necesario por no conocer el espacio de soluciones pero no es nuestro caso.

Lo único que haremos para este algoritmo es asignar los camiones en el horario, de modo que se puedan aplicar todos los operadores disponibles y lo que dejaremos vacío son los camiones, sin ninguna petición asignada. Es posible asignar los camiones de varias formas: tal como vienen, aleatoriamente, o siguiendo algún patrón, en nuestro caso se hará de forma aleatoria. En todos los casos el coste será constante pero aplicar uno u otro puede acercarnos algo a la solución óptima, de todos modos solo existe un paso para aplicar este algoritmo.

**Paso 1:**

Asignar todos los camiones al horario.
*Coste: Θ(6x10) = Θ(1)*

# 5.2. Random

En este caso se trata de colocar los camiones y asignar los pedidos a cada camión de forma aleatoria; siempre vigilando las restricciones del problema.

Iniciar el problema de este modo puede ser adecuado también en problemas en los que no se divisa fácilmente el espacio de soluciones.

**Paso 1:**

Asignar todos los camiones de modo aleatorio.
*Coste: Θ(6x10) = Θ(1)*

**Paso 2:**

Llenar todos los camiones con pedidos aleatorios (de su almacén).
*Coste: max{ Θ(#pedidos) , O(capacidad_total_camiones) } = Θ(n)*

**Coste total:** Θ(1) + Θ (n) = **Θ(n);**

# 5.3. Greedy's

Éste método llamado método avaricioso, intenta acercarse en todo lo posible a una solución óptima (o al menos una buena). Gracias a ésta generación podemos movernos entre espacios de estado próximos al espacio de soluciones.

En éste apartado se indicarán varias opciones para cada paso, lo que hará variar a veces el coste, a veces el espacio necesario, o en ocasiones la heurística de acercarse a la solución. Diferenciaremos entre los dos greedy usados en nuestro sistema.

## 5.3.1 Greedy 1:

**Paso 1:**

Ordenar los pedidos por prioridad (hora entrega, tamaño, beneficio, potenciales perdidas...).
*Coste: HeapSort O(#pedidos log #pedidos) =* ***O(nlogn);*** *no ocupa memoria.*

**Paso 2:**

Recorriendo el horario por centros de producción, ir asignando primero los camiones más grandes, luego los medianos y por último los pequeños. Dejando así en cada centro los camiones más grandes a las primeras horas de la mañana y los más pequeños al final del día.
*Coste: O(6x10) =* ***O(1);***

**Paso 3:**

**A**signar el máximo número de pedidos en el primer camión que quepa, siempre buscando desde el primero que aun tenga espacio.
*Coste: O(#pedidos * 10) =* ***O(n);***

**Coste total:** O(n log n) + O(1) + O(n) = **O(n log n);**

## 5.3.2 Greedy 2:

**Paso 1:**

Ordenar los pedidos por prioridad (hora entrega, tamaño, beneficio, potenciales perdidas...), y al mismo tiempo guardar la suma acumulada de los puntos por almacén. Esto nos servirá para el (Paso 2).

*Coste: HeapSort O(#pedidos log #pedidos) =* ***O(nlogn);*** *no ocupa memoria.*

**Paso 2:**

Mediante la suma acumulada de prioridades del (Paso 1) hacer una ponderación de los camiones y asignar de este modo los grandes medianos y pequeños, siempre con los mayores a primeras horas y los menores a últimas.

*Coste: O(6x10x6) =* ***O(1);***

**Paso 3:**

**A**signar el máximo número de pedidos en el primer camión que quepa, siempre buscando desde el primero que aun tenga espacio.

*Coste: O(#pedidos * 10) =* ***O(n);***

**Coste total:** O(n log n) + O(1) + O(n) = **O(n log n);**

**Alternativa:**

Se puede tener guardado el espacio más grande que existe entre los camiones restantes de ese almacén para no buscar si el pedido ya no cabe en ningún camión existente. También se puede guardar la posición del primer camión no lleno, y el último, para que los recorridos no sean de 10, sino del espacio entre camiones llenos.

Esta alternativa podría acelerar el paso 3 en el caso que existan muchos pedidos y no se pueda llegar a asignar todos. En casos normales en los que el número de pedidos es un número manejable puede que tener que calcular el espacio mínimo en cada iteración ralentice la asignación.

**Conclusión:**

Podemos observar que todos estos algoritmos tienen como coste asintótico O(n log n) donde n es el número de pedidos de la entrada. Haremos uso de estos dos algoritmos ya que producen un coste asumible si tenemos en cuenta que nos acerca lo suficiente a una solución como para evitarnos el coste que conlleva una gran ramificación de los estados.

# 6. Evaluación de resultados (experimentación)

A continuación realizaremos diversos experimentos mediante una serie de escenarios para determinar de qué forma afectan los distintos parámetros que se pueden modificar a las soluciones que intentamos potenciar. Estamos interesados en mejorar varios factores, ya sea maximizar las ganancias, minimizar la diferencia de horas entre las horas límite de entrega y las horas a las que se ha realizado la entrega, o reducir en lo posible el tiempo de ejecución que requiere el análisis de un estado concreto.

## 6.1. Influencia de los operadores

Deberemos determinar qué conjunto de operadores da mejores resultados para una función heurística que optimice el primer criterio de calidad (maximizar las ganancias). Para ello configuraremos un escenario en el que el número peticiones es 250 y la distribución de capacidades de los camiones, pesos de las peticiones y horarios de entrega es equiprobable. Usaremos el algoritmo de Hill Climbing. A partir de éstos resultados deberemos fijar los operadores para el resto de experimentos. Probaremos con todas las estrategias de inicialización de que disponemos.

En un principio se creó un conjunto de operadores que no recorría gran parte del espacio de estados por lo que lo descartamos generando nuevas alternativas que sí cumplían tal función.

**Conjunto Operadores (descartado):**

- Añadir pedido en una posición del horario.
- Quitar pedido de una posición del horario.
- Intercambiar camiones de distinto tamaño.

**Conjunto Operadores 1:**

- Añadir pedido en una posición del horario.
- Quitar pedido de una posición del horario y ponerla en otra hora.
- Intercambiar las horas de entrega de dos pedidos del mismo centro.
- Sustituir un pedido asignado por uno que no lo estaba.
- Intercambiar camiones de distinto tamaño y añadir un pedido en el mayor de ellos.

**Conjunto Operadores 2:**

- Añadir pedido en una posición del horario.
- Quitar pedido en una posición del horario y ponerla en otra hora.
- Intercambiar camiones de distinto tamaño y añadir un pedido en el mayor de ellos.

| | |
|---|---|
| **Observación** | Un conjunto de operadores concreto obtiene mejores resultados. |
| **Planteamiento** | Elegimos distintos conjuntos de operadores y observamos los resultados. |
| **Hipótesis** | Los operadores usados para resolver el problema influyen en el resultado obtenido. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Usamos el algoritmo de Hill Climbing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Experimentamos con las distintas soluciones iniciales (Empty, Random, Greedy 1, Greedy 2).<br>- Experimentamos con los distintos conjuntos de operadores (Operadores 1, Operadores 2).<br>- Hacemos el estudio de los resultados obtenidos (Ganancia). |
| **Resultado** | El conjunto de Operadores 1 obtiene un mejor resultado. |
| **Conclusiones** | El único estado en el que ha influido considerablemente ha sido con la solución inicial Random, con el resto también ha mejorado pero de forma menos notoria. Aún así, los resultados en todos los experimentos favorecen siempre al mismo conjunto de operadores, por tanto se puede decir que la elección de un conjunto de operadores, pese a que no es determinante, sí afecta al resultado final.<br><br>Por otro lado el tiempo de ejecución del segundo grupo de operadores es muy inferior debido, como ya comentamos, a que su ramificación es muy reducida. Pese a esa diferencia y debido a que nuestra prioridad es maximizar las ganancias fijaremos para próximos experimentos el conjunto de operadores 1. |

*T pareada para OP1 - OP2*

| | *N* | *Media* | *Desv.Est.* | *Media del Error estándar* |
|---|---|---|---|---|
| *OP1* | *40* | *92819* | *18488* | *2923* |
| *OP2* | *40* | *88361* | *20044* | *3169* |
| *Diferencia* | *40* | *4458* | *5621* | *889* |

*IC de 95% para la diferencia media:: (2661; 6256)*
*Prueba t de diferencia media = 0 (vs. no = 0): Valor T = 5,02  Valor P = 0,000*

Mediante un estudio con una T-Student apareada para todas las pruebas realizadas sobre el beneficio, podemos observar que el valor P es inferior a 0,05 lo que nos indica que no podemos rechazar la diferencia de resultados entre el operador 1 y el operador 2.

**Tablas utilizadas para el análisis:**

| Empty | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **op1** | 90.695 | 209,2 | 1.186,5 | 130.214,8 | 215,7 | 4.221.467,6 |
| **op2** | 90.009 | 209,3 | 1.207,9 | 96.487,4 | 209,3 | 3.216.079 |

| Random | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **op1** | 94.283 | 223,3 | 816,8 | 29.397,1 | 177,4 | 2.053.825,1 |
| **op2** | 80.838 | 226,3 | 1.033,7 | 7.957 | 125,1 | 653.429,9 |

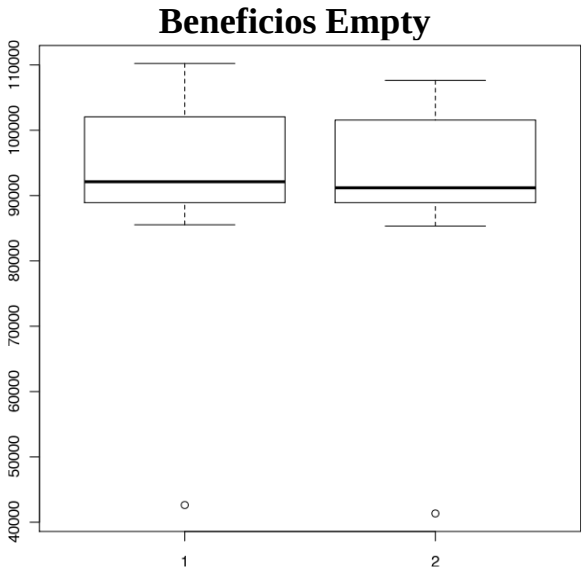
| Greedy1 | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **op1** | 90.512 | 227,9 | 681,5 | 674,4 | 4,4 | 55.254,1 |
| **op2** | 88.718 | 227,7 | 683,3 | 20,3 | 1,3 | 1.971,7 |

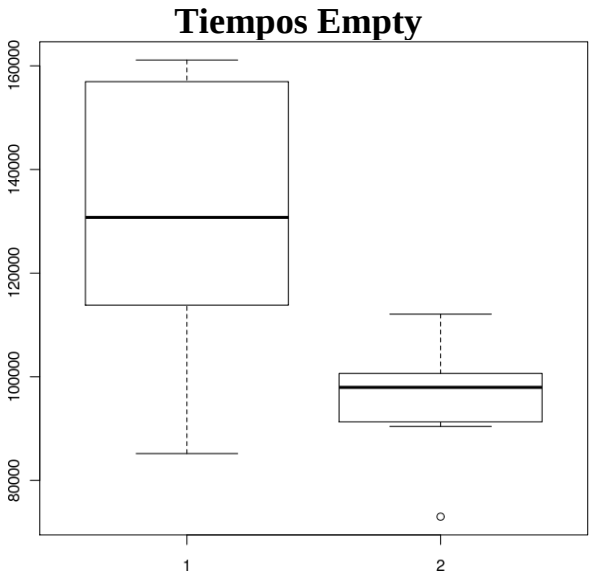
| Greedy2 | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **op1** | 95.785 | 230,4 | 628,4 | 737,2 | 3 | 40.404,2 |
| **op2** | 93.877 | 230,1 | 629,7 | 17 | 0 | 599,9 |

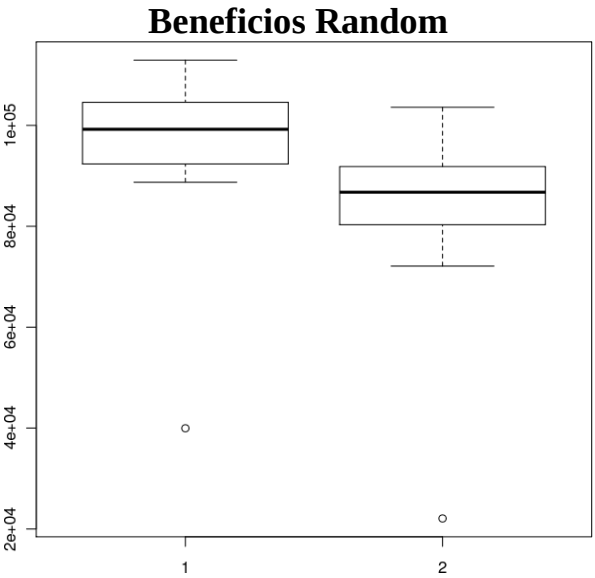
| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **op1** | 92.818,75 | 222,7 | 828,3 | 40.255,87 | 100,12 | 1.592.737,75 |
| **op2** | 88.360,5 | 223,35 | 888,65 | 26.120,42 | 83,92 | 968.020,12 |

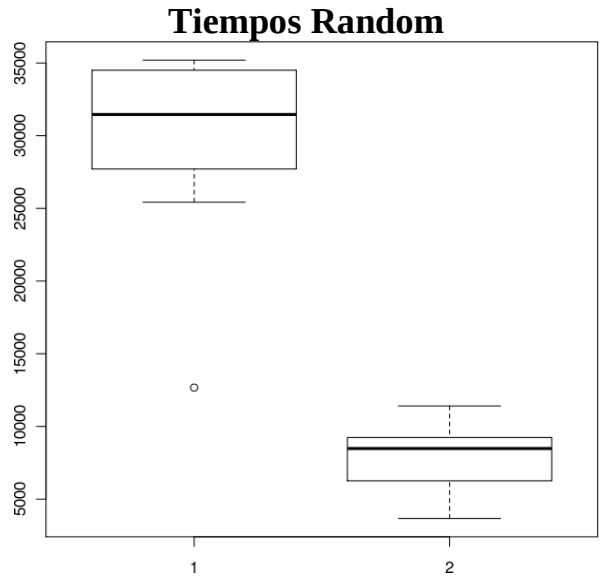
**Gráficos utilizados para el análisis:**

*1 - Conjunto de operadores 1*
*2 - Conjunto de operadores 2*

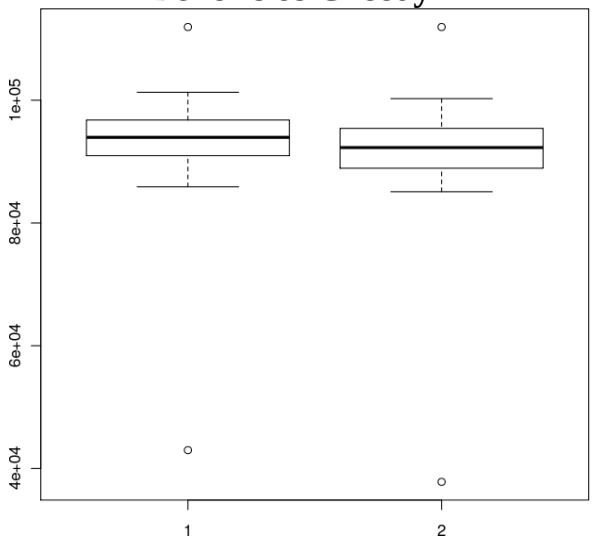


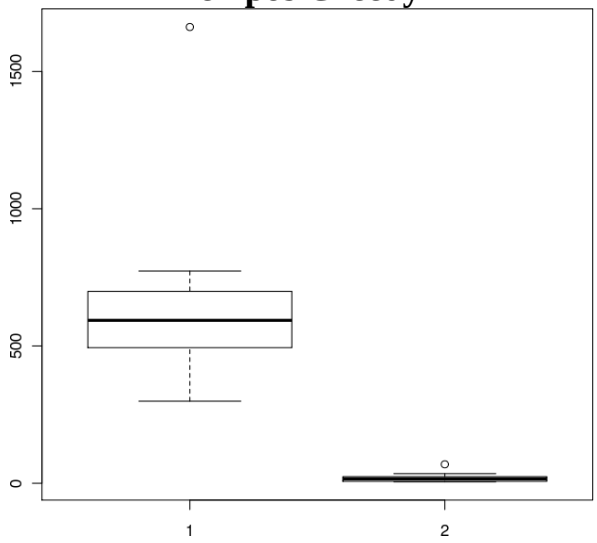



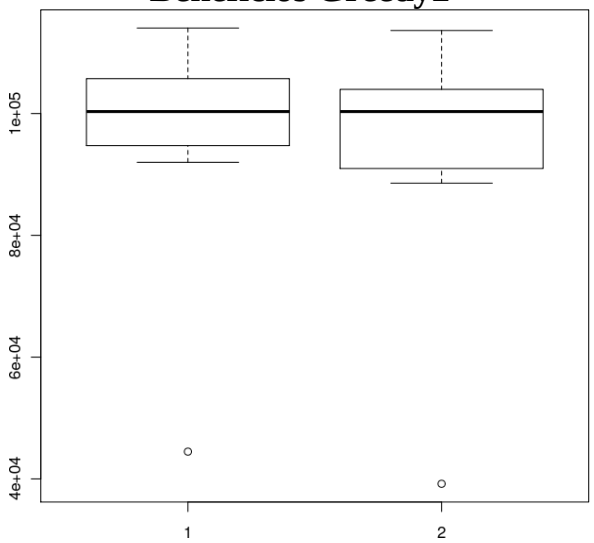
**Beneficios Greedy1**



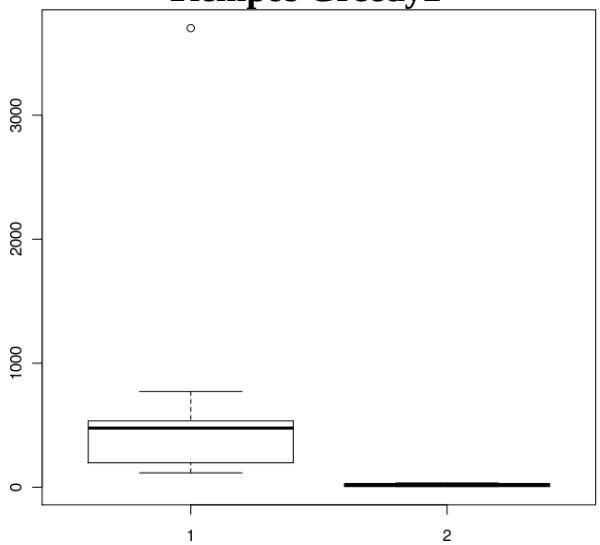

**Tiempos Greedy1**




**Beneficios Greedy2**




**Tiempos Greedy2**



## 6.2. Influencia de la solución inicial

Deberemos determinar qué estrategia de generación de la solución inicial da mejores resultados para la función heurística de maximizar ganancia, con el mismo escenario usado anteriormente (número de peticiones 250, distribución de capacidades de los camiones, pesos de las peticiones y horarios de entrega equiprobables) y usando el algoritmo de Hill Climbing. A partir de estos resultados deberemos fijar la estrategia de generación de la solución inicial para el resto de experimentos.

| | |
|---|---|
| **Observación** | Un método concreto de iniciar el problema obtiene mejores resultados. |
| **Planteamiento** | Elegimos diferentes métodos de iniciar el problema y observamos los resultados. |
| **Hipótesis** | La solución inicial del problema influye en el resultado obtenido. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Usamos el algoritmo de Hill Climbing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Experimentamos con las distintas soluciones iniciales<br>(Empty, Random, Greedy 1, Greedy 2).<br>- Hacemos el estudio de los resultados obtenidos (Ganancia). |
| **Resultado** | La solución inicial Greedy 2 obtiene mejor resultado en un tiempo mucho menor. |
| **Conclusiones** | La diferencia de ganancias obtenida entre las soluciones iniciales no dista mucho una de otra por lo que la elección de una solución inicial concreta no es determinante, se puede ver más abajo un estudio de los beneficios y de las horas en el que nos podemos decantar por Greedy2.<br><br>La diferencia mas significativa se produce en el tiempo de ejecución donde los greedy al generar una solución próxima a la óptima se ahorra mucho tiempo en generación de estados tan solo para acercarse a una solución decente, en el lado opuesto se encuentra el Empty que al partir de un estado vacío debe llenarlo de peticiones antes de empezar a notar que estados son mejores que otros.<br><br>Debido a la notoria diferencia en tiempo y que la ganancia pese a no distar mucho sigue siendo la mejor, fijaremos para futuros experimento la solución inicial generada por el Greedy 2. |

El conflicto respecto al beneficio se puede observar al comparar Random y Greedy 2, para resolverlo hemos preparado una T-Student con muestras apareadas para poder sacar conclusiones:

*T pareada para Random - Greedy2*

| | N | Media | Desv.Est. | Media del Error estándar |
|---|---|---|---|---|
| *Random* | *10* | *94283* | *20512* | *6486* |
| *Greedy2* | *10* | *95785* | *19161* | *6059* |
| *Diferencia* | *10* | *-1502* | *2847* | *900* |

*IC de 95% para la diferencia media:: (-3539; 535)*
*Prueba t de diferencia media = 0 (vs. no = 0): Valor T = -1,67  Valor P = 0,130*

El valor P nos indica que no podemos decir que sean distintas, por lo tanto con los resultados de beneficios no podemos decir que un estado inicial sea mejor que el otro.

Para poder resolver el conflicto vamos a analizar los valores que nos ofrece el segundo criterio que nos indicaba la diferencia de horas absolutas, también mediante una T-Student apareada:

*T pareada para Random_horas - Greedy2_horas*

| | N | Media | Desv.Est. | Media del Error estándar |
|---|---|---|---|---|
| *Random_horas* | *10* | *817* | *357* | *113* |
| *Greedy2_horas* | *10* | *628* | *223* | *71* |
| *Diferencia* | *10* | *188,4* | *158,1* | *50,0* |

*IC de 95% para la diferencia media:: (75,3; 301,5)*
*Prueba t de diferencia media = 0 (vs. no = 0): Valor T = 3,77  Valor P = 0,004*

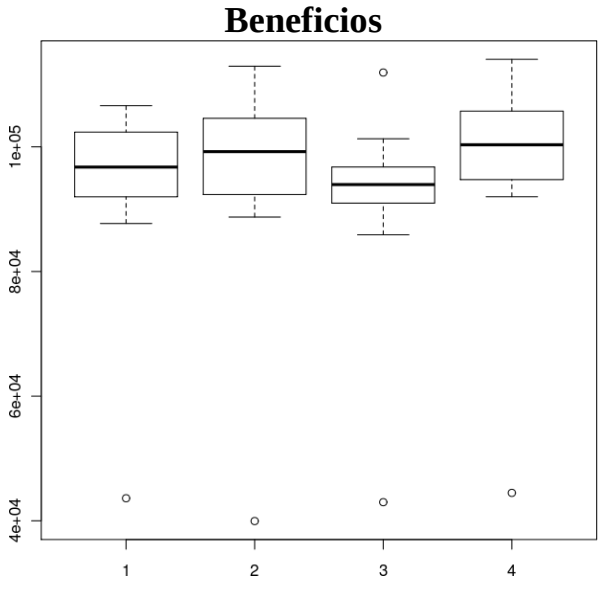
En este caso sí que podemos afirmar que hay una diferencia entre los dos estados iniciales, por lo tanto nos podemos decantar por el inicio Greedy2.

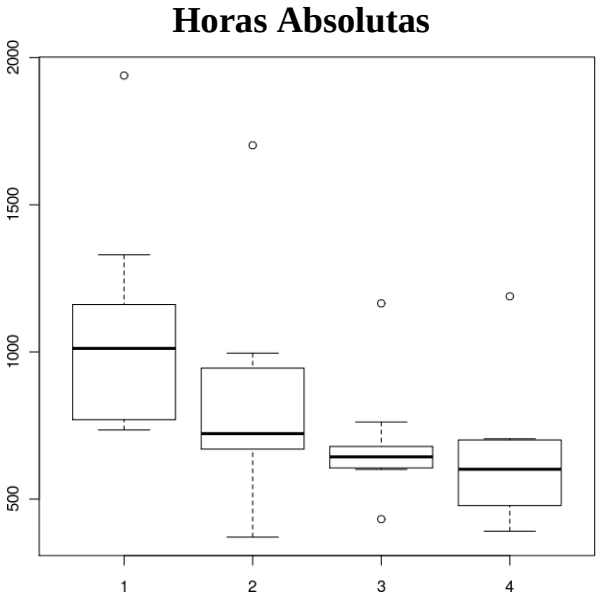
**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **Empty** | 92.468 | 217,1 | 1.070,4 | 137.192,4 | 220,4 | 4.314.804 |
| **Random** | 94.283 | 223,3 | 816,8 | 65.772,5 | 177,4 | 2.053.825,1 |
| **Greedy1** | 90.512 | 227,9 | 681,5 | 1.962 | 4,4 | 55.254,1 |
| **Greedy2** | 95.785 | 230,4 | 628,4 | 1.388,8 | 3 | 40.404,2 |

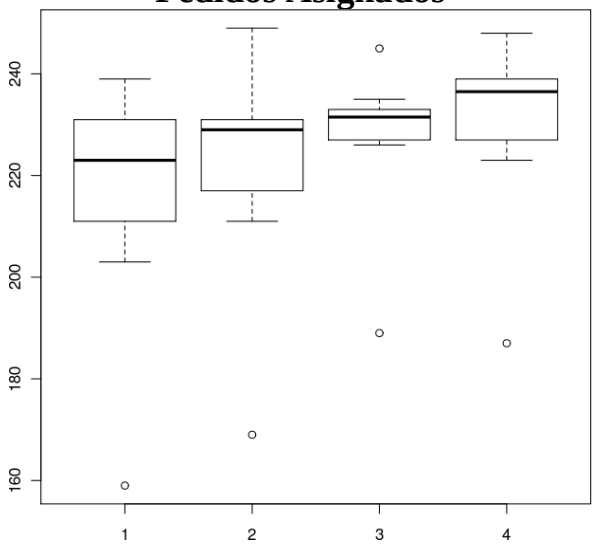
**Gráficos utilizados para el análisis:**

*1 - Empty*
*2 - Random*
*3 - Greedy 1*
*4 - Greedy 2*

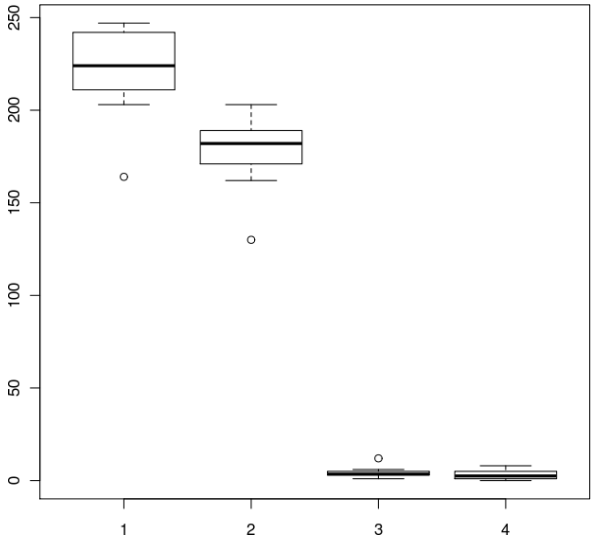


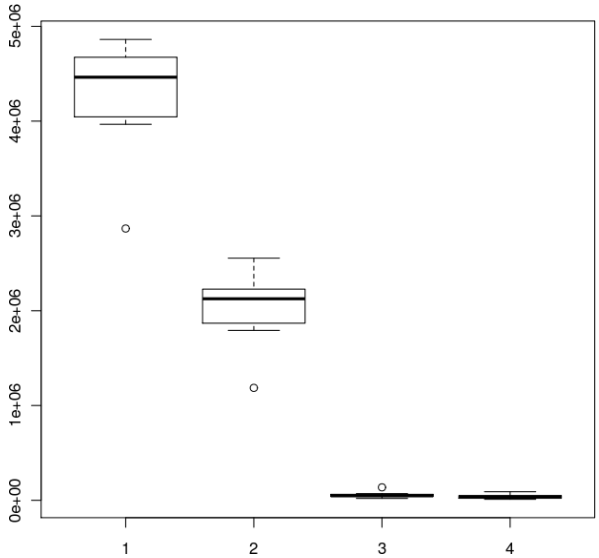
**Pedidos Asignados**



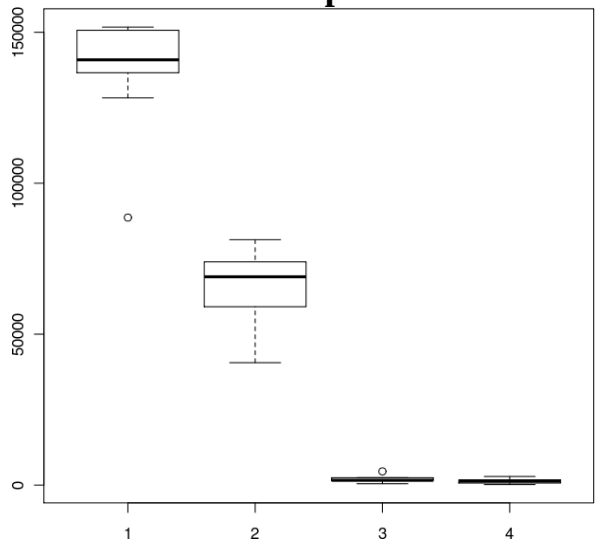

**Pasos**




**Estados Generados**




**Tiempo**



# 6.3. Influencia de los parámetros del Simulated Annealing

Comprobaremos que parámetros dan mejor resultado para el Simulated Annealing con el mismo escenario, usando la función heurística de maximizar ganancia y los operadores y la estrategia de generación de la solución inicial escogidos en los experimentos anteriores (Operadores 1 y Greedy 2).

| | |
|---|---|
| **Observación** | Puede que algunos parámetros de la función de Simulated Annealing ofrezcan mejores resultados. |
| **Planteamiento** | Realizamos cuatro experimentos individuales, uno por cada parámetro (K, lambda, Stiter y Steps) y evaluamos los resultados. |
| **Hipótesis** | Modificar los valores debe variar los beneficios obtenidos. |
| **Método** | - Modificar el parámetro K asignándole los valores<br>{ 5 , 10 , 15 , 20 , 25 , 30 }<br>- Modificar el parámetro Lambda asignándole los valores :<br>{ 0.001 , 0.005 , 0.01 , 0.05 , 0.1 , 0.5 }<br>- Modificar el parámetro Lambda asignándole los valores :<br>{ 0.001 , 0.005 , 0.01 , 0.05 , 0.1 , 0.5 }<br>- Modificar el parámetro Steps asignándole los valores :<br>{ 1000 , 5000 , 10000 , 20000 , 40000 , 60000 , 90000 }<br>- Modificar el parámetro Stiter asignándole los valores :<br>{ 10 , 25 , 50 , 75 , 100 , 150 , 200 }<br><br>( Los experimentos individuales están explicados en cada subapartado. ) |
| **Resultado** | No se aprecia una clara variación de beneficios. |
| **Conclusiones** | La diferencia de valores para los parámetros han modificado más el tiempo de ejecución que los beneficios observados. De todos modos hemos seleccionado aquellos valores que maximizaban el beneficio, ya que los tiempos de cómputo nunca han sido excesivos. |

## Parámetro K

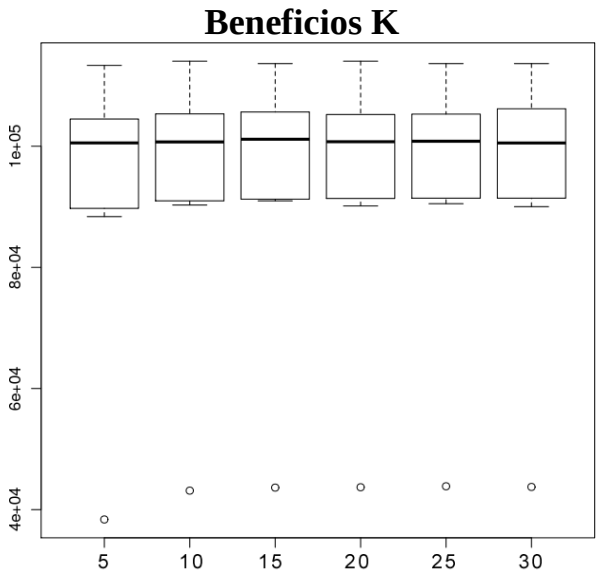
Hace que se generen muchos sucesores cuando éste se aumenta, y por eso la ejecución es más lenta, y entonces el límite establecido puede hacer que se pare antes de encontrar una solución buena.

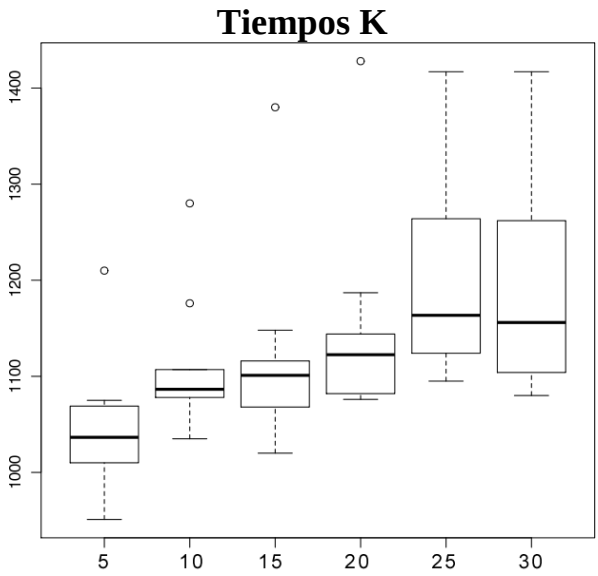
| | |
|---|---|
| **Observación** | Puede que modificando el valor del parámetro K de la función de Simulated Annealing obtengamos mejores resultados respecto a beneficios. |
| **Planteamiento** | Escogemos distintos valores para K i comparamos los resultados obtenidos. |
| **Hipótesis** | Modificar los valores varía el valor de los beneficios obtenidos. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Elegimos los parámetros (Steps: 10000, Stiter: 100, Lambda: 0.05)<br>- Experimentamos con el parámetro K asignándole los valores<br>{ 5 , 10 , 15 , 20 , 25 , 30 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Ganancia y Tiempo de ejecución). |
| **Resultado** | No se aprecia una clara variación de beneficios. Por lo contrario, si que se aprecia un aumento del tiempo de cómputo. |
| **Conclusiones** | El valor de K no ofrece una clara mejoría de la solución para los operadores y el inicio greedy que hemos usado, de todos modos ponderando el tiempo y los beneficios nos hemos decantado por el valor 15 ya que ofrece la mejor media y el coste temporal varía en menos de 1 segundo entre muestras. |

**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **5** | 93.842 | 232 | 545 | 1.043 | 373 | 112.512 |
| **10** | 95.205 | 231 | 607 | 1.107 | 1.772 | 113.137 |
| **15** | 95.487 | 231 | 606 | 1.116 | 1.557 | 113.496 |
| **20** | 95.276 | 231 | 600 | 1.150 | 1.194 | 113.063 |
| **43** | 95.076 | 231 | 617 | 1.197 | 1.473 | 113.200 |
| **30** | 95.383 | 231 | 604 | 1.189 | 1.356 | 113.106 |

**Gráficos utilizados para el análisis:**



## Parámetro Lambda

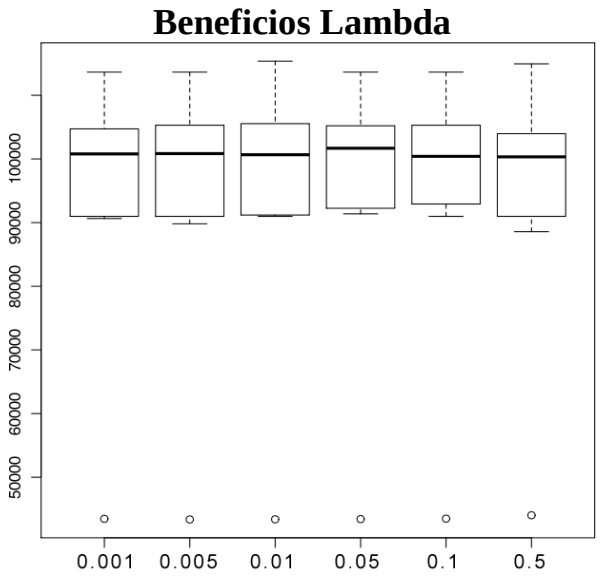
Aumentar lambda excesivamente hace que la ejecución acabe demasiado pronto dando un resultado bastante malo. En cambio, al reducirla mucho (casi a 0), el programa puede encontrar una solución buena aunque tarde demasiado.

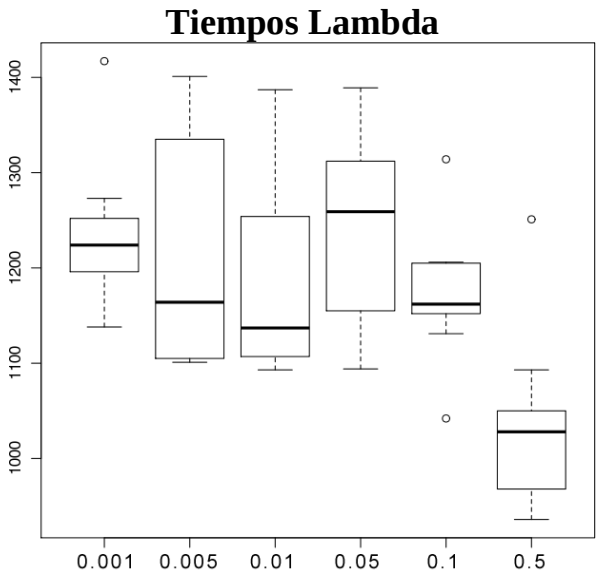
| | |
|---|---|
| **Observación** | Puede que modificando el valor del parámetro Lambda de la función de Simulated Annealing obtengamos mejores resultados respecto a los beneficios. |
| **Planteamiento** | Escogemos distintos valores para Lambda y comparamos los resultados obtenidos. |
| **Hipótesis** | Modificar los valores varía el valor de los beneficios obtenidos. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Elegimos los parámetros (Steps: 10000, Stiter: 100, K: 15)<br>- Experimentamos con el parámetro Lambda asignándole los valores<br>{ 0.001 , 0.005 , 0.01 , 0.05 , 0.1 , 0.5 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Ganancia y Tiempo de ejecución). |
| **Resultado** | No se aprecia una clara variación de beneficios. |
| **Conclusiones** | La variación del valor de Lambda no ofrece una clara mejoría de la solución para los operadores y el inicio greedy que hemos usado, de todos modos el valor 0.05 ofrece la mejor media en ganancia y el coste temporal varía en menos de 1 segundo entre las distintas muestras estudiadas. |

**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **0,001** | 94.953 | 231 | 614 | 1.230 | 1.304 | 113.114 |
| **0,005** | 95.191 | 231 | 608 | 1.205 | 1.273 | 113.046 |
| **0,010** | 95.649 | 232 | 599 | 1.173 | 1.544 | 113.011 |
| **0,050** | 95.844 | 232 | 599 | 1.239 | 1.552 | 113.066 |
| **0,100** | 95.210 | 231 | 611 | 1.170 | 1.260 | 113.059 |
| **0,500** | 94.588 | 230 | 621 | 1.037 | 107 | 112.495 |

**Gráficos utilizados para el análisis:**



## Parámetro Stiter

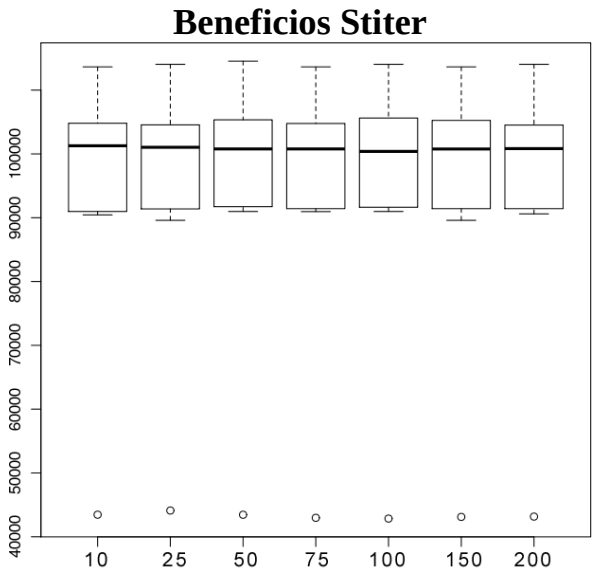
El parámetro stiter indica el número de iteraciones que realizará por paso de temperatura, no influye mucho pero con valores pequeños suele dar peores resultados.

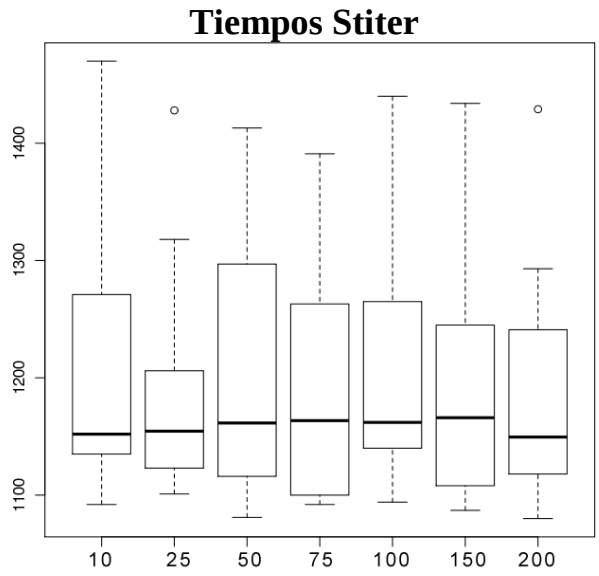
| | |
|---|---|
| **Observación** | Puede que modificando el valor del parámetro Stiter de la función de Simulated Annealing obtengamos mejores resultados respecto a beneficios. |
| **Planteamiento** | Escogemos distintos valores para Stiter y comparamos los resultados obtenidos. |
| **Hipótesis** | Modificar los valores varía el valor de los beneficios obtenidos. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Elegimos los parámetros (Steps: 10000, K: 15, Lambda: 0.05)<br>- Experimentamos con el parámetro Stiter asignándole los valores<br>{ 10 , 25 , 50 , 75 , 100 , 150 , 200 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Ganancia y Tiempo de ejecución). |
| **Resultado** | No se aprecia una clara variación de beneficios. |
| **Conclusiones** | El valor de Stiter no hace mejorar de forma notable los beneficios, tampoco el tiempo se ve afectado. Escogemos el que ofrece una mejor media de beneficios, en este caso con un stiter de 100. |

**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **10** | 95.077 | 231 | 607 | 1.201 | 1.290 | 112.731 |
| **25** | 95.106 | 231 | 598 | 1.190 | 1.234 | 113.307 |
| **50** | 95.300 | 231 | 603 | 1.195 | 1.718 | 113.421 |
| **75** | 95.043 | 231 | 609 | 1.191 | 1.350 | 112.712 |
| **100** | 95.348 | 231 | 601 | 1.200 | 1.371 | 113.815 |
| **150** | 94.988 | 231 | 606 | 1.192 | 1.335 | 112.875 |
| **200** | 95.121 | 231 | 611 | 1.193 | 1.865 | 113.409 |

**Gráficos utilizados para el análisis:**



## Parámetro Steps

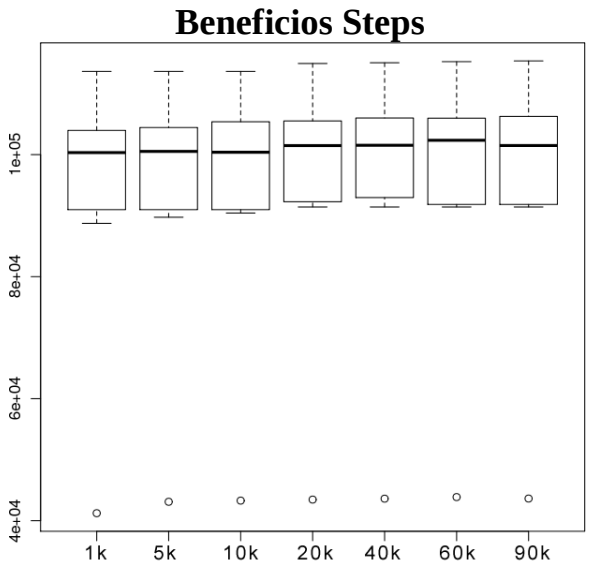
Los steps sirven para poner un límite a la ejecución. Con pocos estados, el límite puede ser bajo y hacer que la ejecución sea muy rápida, en cambio con muchos estados hay que aumentar el límite para que sea capaz de encontrar una buena solución.

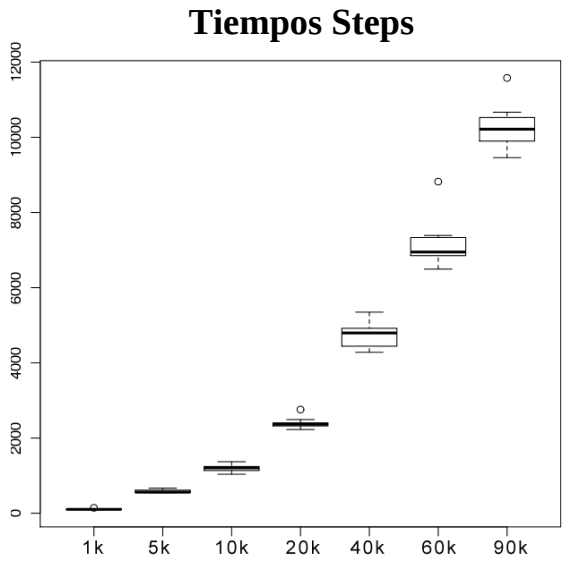
| | |
|---|---|
| **Observación** | Puede que modificando el valor del parámetro Steps de la función de Simulated Annealing obtengamos mejores resultados respecto a beneficios. |
| **Planteamiento** | Escogemos distintos valores para Steps y comparamos los resultados obtenidos. |
| **Hipótesis** | Modificar los valores varía el valor de los beneficios obtenidos. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Elegimos los parámetros (Stiter: 100, K:15, Lambda: 0.05)<br>- Experimentamos con el parámetro Steps asignándole los valores<br>{ 1000 , 5000 , 10000 , 20000 , 40000 , 60000 , 90000 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Ganancia y Tiempo de ejecución). |
| **Resultado** | No se aprecia una clara variación de beneficios. En cambio el tiempo crece exponencialmente. |
| **Conclusiones** | El valor de Steps tampoco ofrece grandes cambios en el resultado pero el valor máximo de las medias pertenece a 60.000 pasos. El tiempo que tarda en hacer todos los pasos indicados crece de forma lineal (en el gráfico parece exponencial por los valores escogidos para las muestras). |

**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **1000** | 94.096 | 230 | 630 | 108 | 29 | 11.078 |
| **5000** | 94.770 | 231 | 621 | 581 | 528 | 56.260 |
| **10000** | 95.131 | 231 | 610 | 1.203 | 1.458 | 112.725 |
| **20000** | 95.843 | 232 | 590 | 2.394 | 3.405 | 225.959 |
| **40000** | 96.026 | 232 | 596 | 4.743 | 5.494 | 454.891 |
| **60000** | 96.178 | 232 | 593 | 7.134 | 4.526 | 680.938 |
| **90000** | 96.005 | 232 | 595 | 10.263 | 8.280 | 1.019.754 |

**Gráficos utilizados para el análisis:**



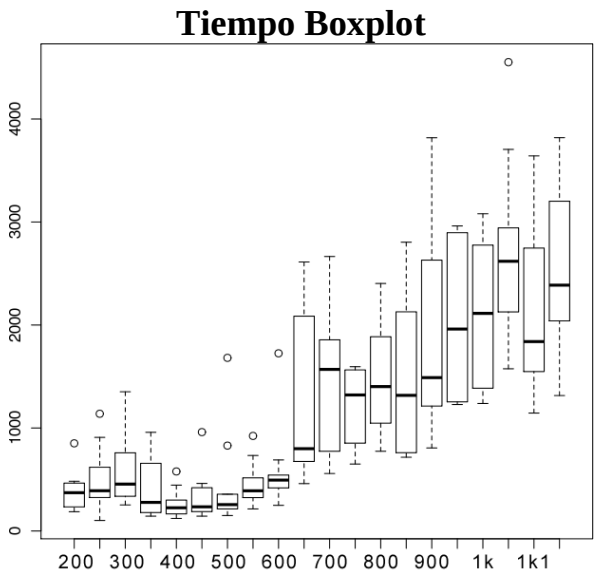



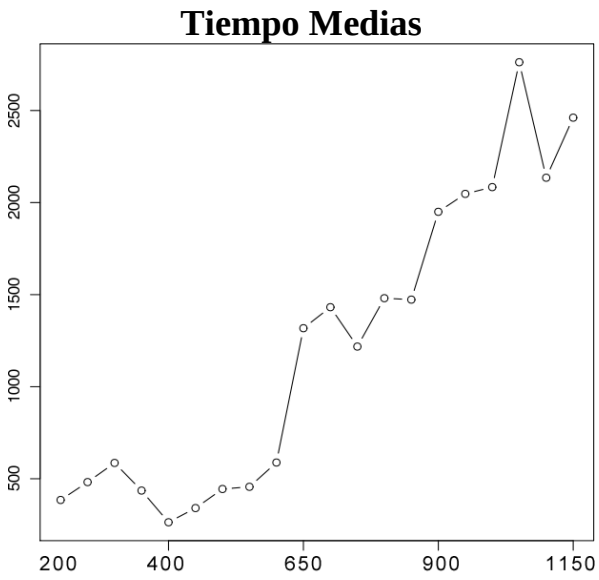
## 6.4. Influencia del número de peticiones

Con el escenario de los apartados anteriores, estudiaremos como evoluciona el tiempo de ejecución para hallar la solución en función del número de peticiones que tenemos. Para ello empezaremos con 200 peticiones e iremos aumentándolas de 50 en 50 hasta ver la tendencia. Usaremos el algoritmo de Hill Climbing y la heurística de maximizar ganancias.

| | |
|---|---|
| **Observación** | Aumentando el número de peticiones se podrá apreciar el tipo de crecimiento en el tiempo de ejecución. |
| **Planteamiento** | Vamos aumentando el número de peticiones y estudiamos los resultados. |
| **Hipótesis** | Aumentando los pedidos aumentará el tiempo de cálculo. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Experimentamos con el número de peticiones en saltos de 50 desde 200 hasta 1150<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Tiempo de ejecución). |
| **Resultado** | Al principio con pasos inferiores a 600 el tiempo de ejecución es algo oscilante, pero a partir de este valor se observa un crecimiento totalmente lineal. |
| **Conclusiones** | A medida que se aumentan los pedidos también aumenta el número de estados posibles que se pueden generar. Aunque se podría pensar que el crecimiento podría llegar a ser exponencial, las restricciones de nuestros operadores no permiten que el problema “explote” manteniendo una solución óptima conseguida a través de la buena generación del estado inicial. |

**Gráficos utilizados para el análisis:**



## 6.5. Influencia de la función heurística

Con el escenario del primer apartado y con 200, 250 y 300 peticiones, estimaremos la diferencia entre la ganancia obtenida y el tiempo de ejecución para hallar la solución, usando las dos heurísticas (maximizar ganancia y minimizar diferencia de horas). Usaremos para ello el algoritmo de Hill Climbing.

| | |
|---|---|
| **Observación** | Aplicando distintos heurísticos se observaran distintos tipos de ganancias; por beneficios o por diferencia de horas absolutas. |
| **Planteamiento** | Vamos a hacer las pruebas apareadas por número de peticiones. |
| **Hipótesis** | El heurístico encargado de Maximizar beneficios dará mayores beneficios que el de Minimizar la diferencia de horas absolutas, y este último dará menores diferencias en las horas de entrega. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Hill Climbing.<br>- Experimentamos con 200, 250 y 300 peticiones.<br>- Experimentamos cada escenario primero Maximizando Beneficios y luego Minimizando la diferencia de horas absolutas.<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Beneficios, Horas absolutas y Tiempo de ejecución). |
| **Resultado** | Se puede apreciar que realmente hay una diferencia en el calculo de horas absolutas entre los dos heurísticos, al contrario en los beneficios se puede apreciar muy poca ventaja por parte del que los intenta Maximizar.<br>Los tiempos de ejecución son mayores para el heurístico de las horas absolutas. |
| **Conclusiones** | A medida que se satura el estado inicial la diferencia de horas absolutas es menor ya que el número de pedidos no asignados crece y hace aumentar este valor por igual. En los tiempos se puede deducir que nuestro estado inicial está pensado exclusivamente para aumentar los beneficios, lo que hace que con el primer heurístico no se encuentren soluciones más óptimas, en cambio con el segundo heurístico sí.<br><br>También se puede observar que el heurístico que minimiza las horas logra mejorar tanto la diferencia de horas como el beneficio, eso es debido a que reducir el margen de horas también logra que se maximicen los beneficios, pese a ello, no logra tan buenos resultados como el heurístico específico para ese objetivo, el de maximizar el beneficio. |

**Tablas utilizadas para el análisis:**

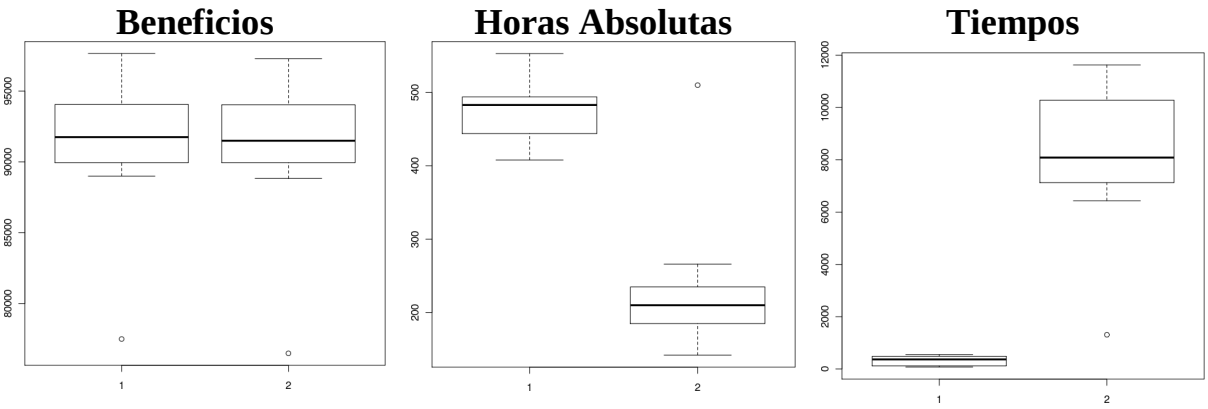
| Media | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **Max Beneficios** | 85.286 | 222,83 | 758,10 | 468,43 | 2,03 | 32.674,20 |
| **Min Horas** | 84.485,67 | 223,33 | 645,77 | 4.444,73 | 43,67 | 389.378,70 |

| Desviación | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **Max Beneficios** | 21.734,65 | 23,58 | 378,80 | 330,97 | 2,39 | 27.939,28 |
| **Min Horas** | 22.696,06 | 24,01 | 460,98 | 3.693,78 | 44,39 | 343.885,84 |

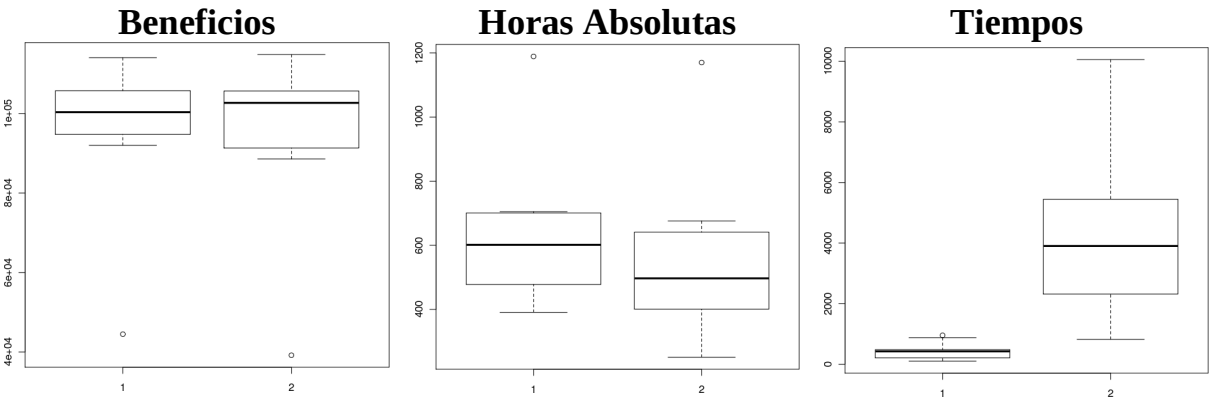
**Gráficos utilizados para el análisis:**

*1 - Maximizar Beneficios*
*2 - Minimizar Horas Absolutas*

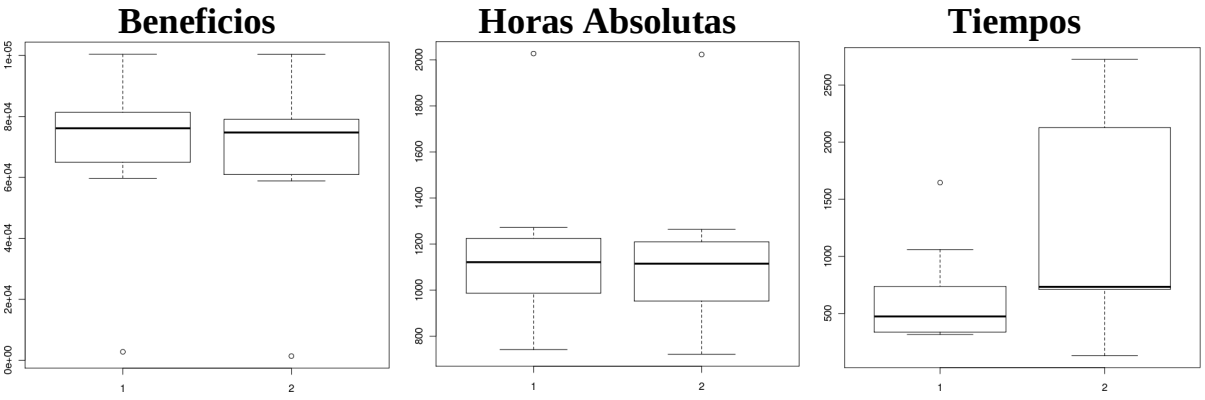
## 200 Pedidos




## 250 Pedidos




## 300 Pedidos



# 6.6. Comparación de algoritmos

Dados los mismos escenarios que en el apartado anterior, estimaremos la diferencia entre la ganancia obtenida con el Hill Climbing y el Simulated Annealing para las dos heurísticas (maximizar ganancia y minimizar diferencia de horas) y el tiempo de ejecución para hallar la solución de ambos algoritmos.

Primero repetiremos el experimento anterior que comparaba los diferentes heurísticos pero esta vez con el algoritmo del Simulated Annealing y con los parámetros que establecimos como óptimos en uno de los apartados anteriores.

| | |
|---|---|
| **Observación** | Aplicando distintos heurísticos se observaran distintos tipos de ganancias; por beneficios o por diferencia de horas absolutas. |
| **Planteamiento** | Vamos a hacer las pruebas apareadas por número de peticiones. |
| **Hipótesis** | El heurístico encargado de Maximizar beneficios dará mayores beneficios que el de Minimizar la diferencia de horas absolutas, y este último dará menores diferencias en las horas de entrega. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Simulated Annealing<br>- Elegimos los parámetros para el Simulated Annealing<br>(Steps: 60000, Stiter: 100, K: 15, Lambda: 0.05)<br>- Experimentamos con 200, 250 y 300 peticiones.<br>- Experimentamos cada escenario primero Maximizando Beneficios y luego Minimizando la diferencia de horas absolutas.<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Beneficios, Horas absolutas y Tiempo de ejecución). |
| **Resultado** | Se puede apreciar que realmente hay una diferencia en el cálculo de horas absolutas entre los dos heurísticos, al contrario en los beneficios se puede apreciar muy poca ventaja por parte del que los intenta Maximizar.<br>Los tiempos de ejecución son muy variables. |
| **Conclusiones** | Los beneficios que se consiguen no varían mucho entre los dos heurísticos |

**Tablas utilizadas para el análisis:**

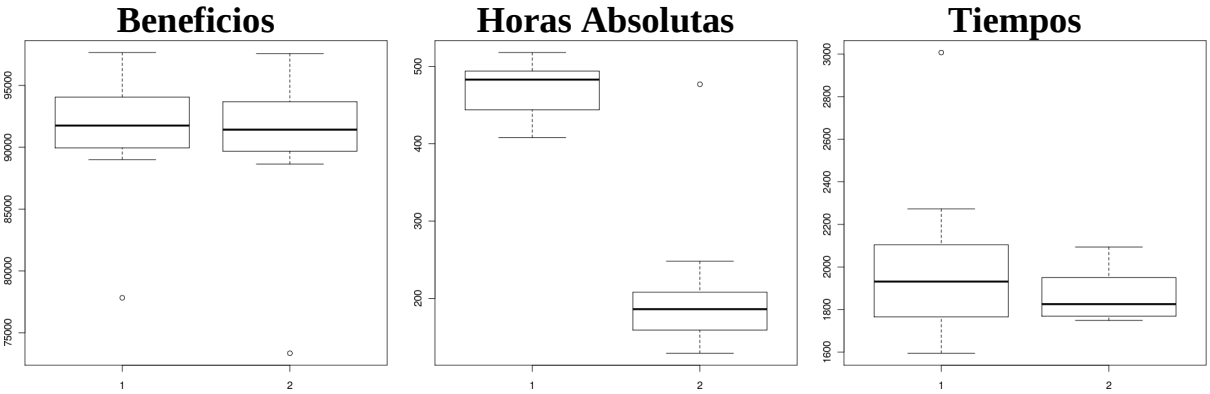
| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **Max Beneficios** | 84.880,33 | 223,43 | 745,47 | 2,451,07 | 1.627,20 | 223.166,83 |
| **Min Horas** | 81.648,33 | 225,63 | 606,00 | 2.411,77 | 4.161,80 | 225.139,97 |

| Desviación | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **Max Beneficios** | 22.086,95 | 23,84 | 380,60 | 436,15 | 1.908,78 | 41.170,39 |
| **Min Horas** | 23.148,45 | 24,37 | 440,96 | 457,15 | 533,42 | 32.804,30 |

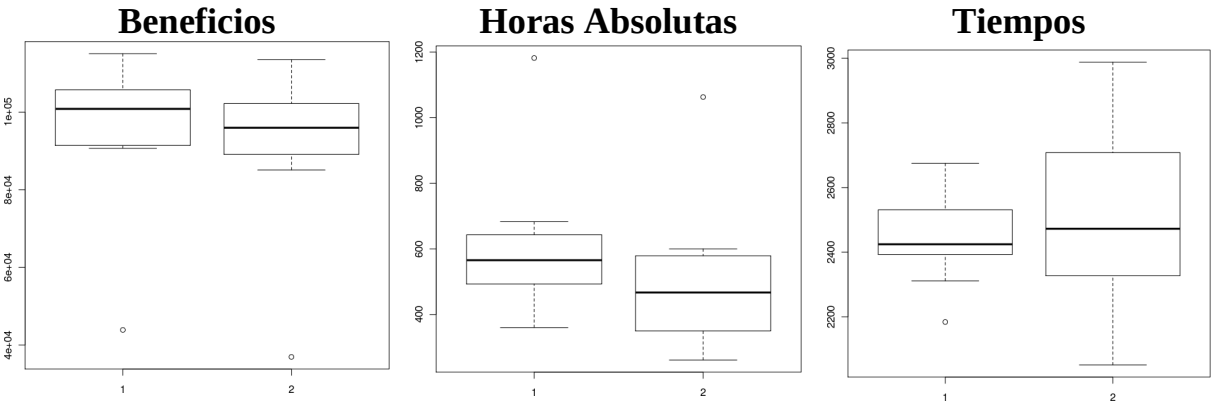
**Gráficos utilizados para el análisis:**

*1 - Maximizar Beneficios*
*2 - Minimizar Horas Absolutas*

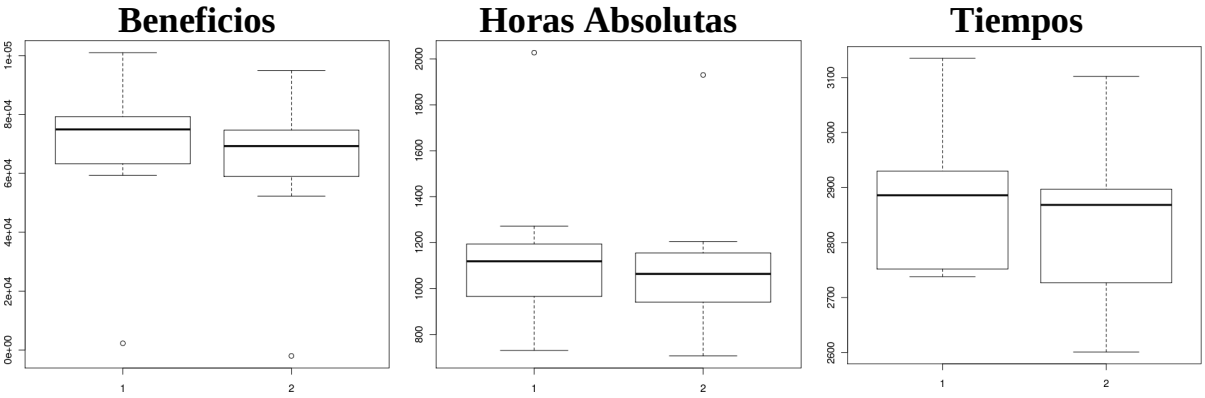
## Simulated Annealing 200 Pedidos




## Simulated Annealing 250 Pedidos




## Simulated Annealing 300 Pedidos



Ahora compararemos las diferencias que hay entre los resultados obtenidos por el algoritmo de Hill Climbing y el de Simulated Annealing.

| | |
|---|---|
| **Observación** | Queremos ver si existen diferencias entre Hill Climbing y Simulated Annealing. |
| **Planteamiento** | Vamos a coger los resultados obtenidos anteriormente por cada función y compararemos los resultados entre Hill Climbing y Simulated Annealing. |
| **Hipótesis** | Los dos algoritmos dan los mismos resultados o hay uno mejor que otro. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Elegimos los parámetros para el Simulated Annealing<br>(Steps: 60000, Stiter: 100, K: 15, Lambda: 0.05)<br>- Experimentamos con 200, 250 y 300 peticiones.<br>- Experimentamos con los algoritmos de Hill Climbing y Simulated Annealing<br>- Experimentamos cada escenario primero Maximizando Beneficios y luego Minimizando la diferencia de horas absolutas.<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Beneficios, Horas absolutas y Tiempo de ejecución). |
| **Resultado** | La diferencia en los beneficios y en las horas absolutas es ínfima, abajo podemos ver un estudio de los resultados. Por el contrario podemos ver que el tiempo que tardan en encontrar la solución, en el caso de Hill Climbing para Maximizar Beneficios es superior al de Minimizar diferencia de horas absolutas, quedando reflejado en la desviación estándar del tiempo, en cambio el Simulated Annealing tiene una distribución más monótona. |
| **Conclusiones** | Observando los resultados podemos ver que aunque el Hill Climbing genere todos los estados sucesores y entre ellos elija el mejor, no dista mucho de la solución que ofrece un Simulated Annealing que genera estados aleatorios y sigue unas probabilidades de elección de sucesores. Todo esto hace que el Hill Climbing pueda crecer en tiempo según el número de sucesores, mientras el Simulated Annealing siempre va a tardar unos tiempos similares (esto se puede ver en la desviación de los tiempos). |

Respecto a los beneficios no podemos ver la diferencia, por lo que vamos a estudiar una T-Student apareada por si con ella podemos sacar alguna conclusión:

*T pareada para HC_ben - SA_ben*

| | N | Media | Desv.Est. | Media del Error estándar |
|---|---|---|---|---|
| *HC_ben* | *30* | *85286* | *21735* | *3968* |
| *SA_ben* | *30* | *84880* | *22087* | *4033* |
| *Diferencia* | *30* | *406* | *1333* | *243* |

*IC de 95% para la diferencia media:: (-92; 903)*
*Prueba t de diferencia media = 0 (vs. no = 0): Valor T = 1,67  Valor P = 0,106*

Con el P-value tan grande no podemos decir que son distintas, por lo tanto respecto a los beneficios no hay diferencia alguna.

Vamos a comparar los datos que nos ofrecen respecto a las horas absolutas, mediante el heurístico de minimizar-las.

*T pareada para HC_horas - SA_horas*

| | N | Media | Desv.Est. | Media del Error estándar |
|---|---|---|---|---|
| *HC_horas* | *30* | *645,8* | *461,0* | *84,2* |
| *SA_horas* | *30* | *606,0* | *441,0* | *80,5* |
| *Diferencia* | *30* | *39,77* | *28,82* | *5,26* |

*IC de 95% para la diferencia media:: (29,00; 50,53)*
*Prueba t de diferencia media = 0 (vs. no = 0): Valor T = 7,56  Valor P = 0,000*

En este caso si que podemos decir que son distintas ya que el P-value es 0 lo que nos haría decantar por el SA que da mejores resultados para minimizar la diferencia de horas absolutas.

**Tablas utilizadas para el análisis:**

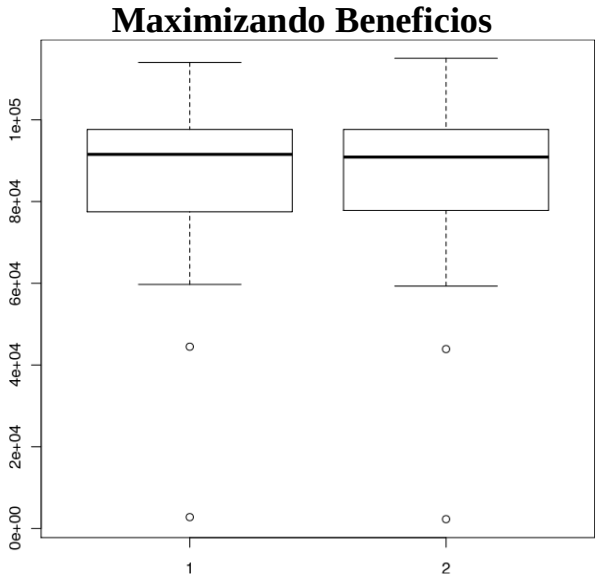
| **Media** | **Beneficios** | **Asign.** | **Horas** | **Tiempo (ms)** | **Pasos** |
|---|---|---|---|---|---|
| **HC** | 84.885,83 | 223,08 | 701,93 | 2.456,58 | 22,85 |
| **SA** | 83.264,33 | 224,53 | 675,73 | 2.431,42 | 2.894,50 |

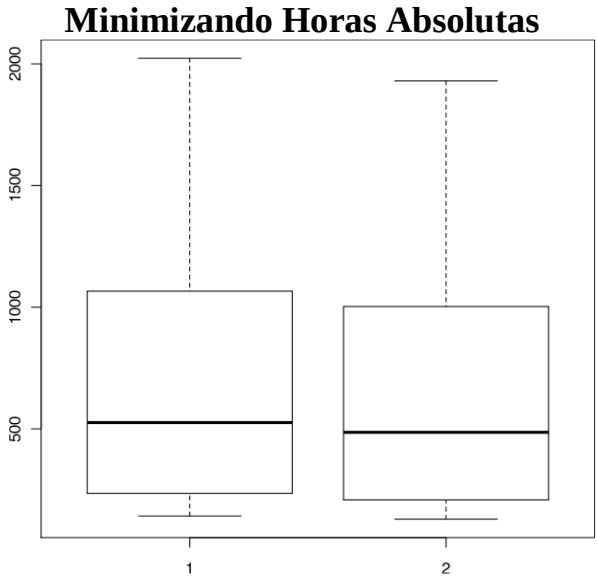
| **Media** Desviaciones | **Beneficios** | **Asign.** | **Horas** | **Tiempo (ms)** | **Pasos** |
|---|---|---|---|---|---|
| **HC** | 22.215,35 | 23,79 | 419,89 | 2.012,37 | 23,39 |
| **SA** | 22.617,70 | 24,10 | 410,78 | 446,65 | 1.221,10 |

**Gráficos utilizados para el análisis:**

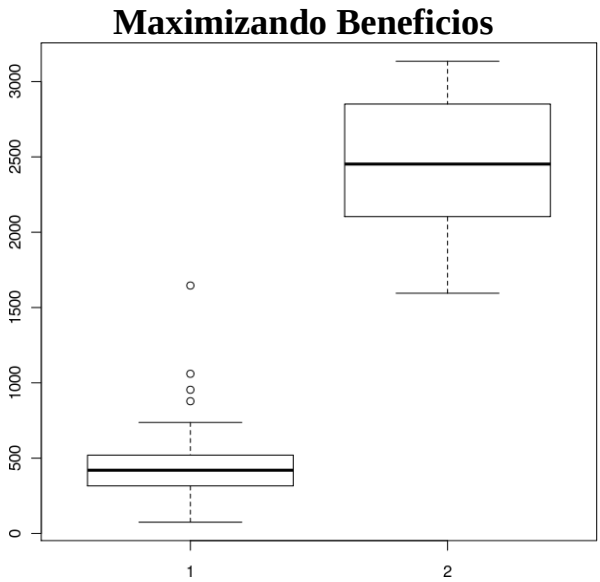
*1 - Hill Climbing*
*2 - Simulated Annealing*

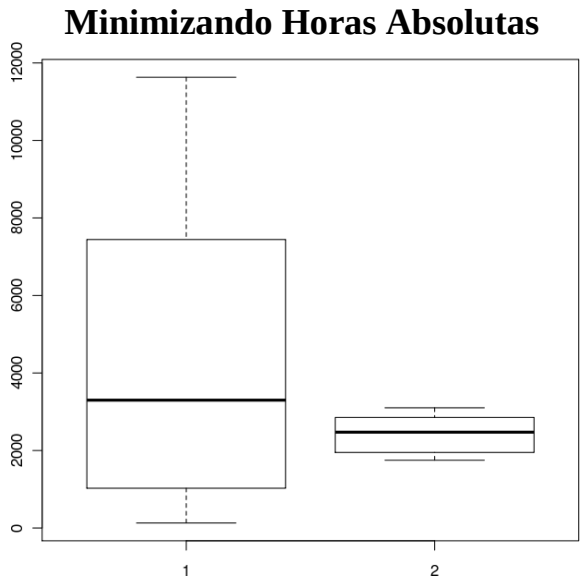
## Beneficios y Horas Absolutas




## Tiempo



# 6.7. Influencia de la flota de camiones

Para solucionar el problema podemos disponer de diferentes flotas de camiones. Dado el escenario del primer experimento, donde los pesos de las peticiones y horarios de entrega se distribuyen de forma equiprobable, y con una cantidad de peticiones de 250, probaremos tres variaciones en las que tengamos un 50 % de camiones de un tamaño y un 25 % de cada uno de los otros dos. Usaremos el heurístico de maximizar ganancias y el algoritmo de Hill Climbing. Explicaremos la diferencia entre las soluciones de los distintos escenarios.

| | |
|---|---|
| **Observación** | Queremos observar como varían los resultados si en nuestra flota de camiones disponibles se modifica la cantidad de cada tipo de camión. |
| **Planteamiento** | Probaremos de encontrar soluciones para una distribución que tenga el doble de camiones pequeños, otra con el doble de medianos y por último una con el doble de grandes. |
| **Hipótesis** | Cuando aumenta el espacio disponible en los camiones mejoran los resultados. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones<br>- Elegimos las horas y pesos de las peticiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Hill Climbing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Experimentamos con 3 escenarios en función del tipo de camiones:<br>{ 30 , 15 , 15 } { 15 , 30 , 15 } y { 15 , 15 , 30 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Beneficios, Horas absolutas, Tiempo de ejecución y número de pedidos asignados). |
| **Resultado** | Cuanto mayor es la capacidad total mayores son los beneficios, se consiguen entregar más pedidos y disminuye la diferencia de horas de entrega. |
| **Conclusiones** | Podemos concluir que con el aumento de capacidad de los camiones podemos entregar más pedidos, con todos los beneficios que ello conlleva, más ganancias y entregas más puntuales. |

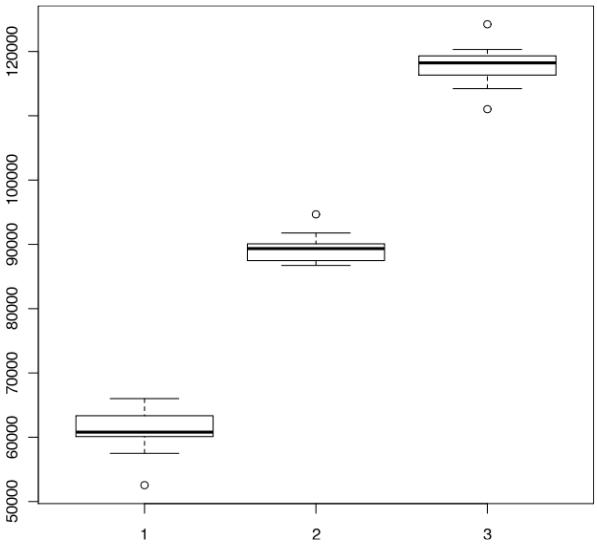
**Tablas utilizadas para el análisis:**

| Medias | Beneficios | Asign. | Horas | Tiempo (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **30 , 15 , 15** | 60.836 | 200,6 | 981,9 | 1.674,3 | 3,2 | 37.373,6 |
| **15 , 30 , 15** | 89.497 | 221,5 | 643,5 | 1.745,4 | 3,1 | 40.242,2 |
| **15 , 15 , 30** | 117.851 | 249,1 | 400,9 | 893,1 | 0,5 | 17.312,9 |

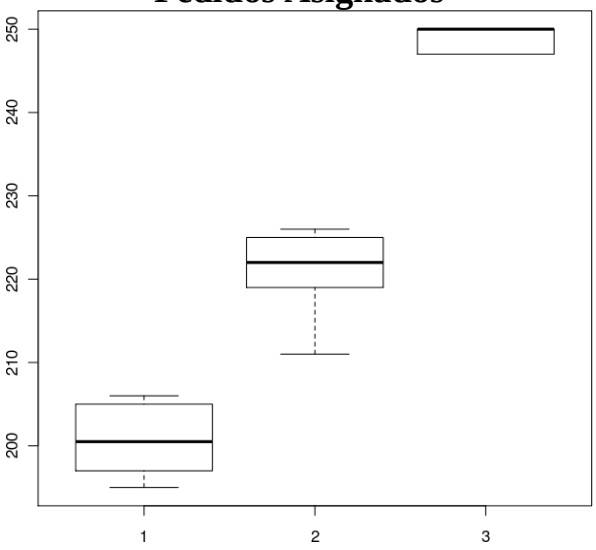
**Gráficos utilizados para el análisis:**

*1 - 30 Pequeños 15 Medianos 15 Grandes*
*2 - 15 Pequeños 30 Medianos 15 Grandes*
*3 - 15 Pequeños 15 Medianos 30 Grandes*

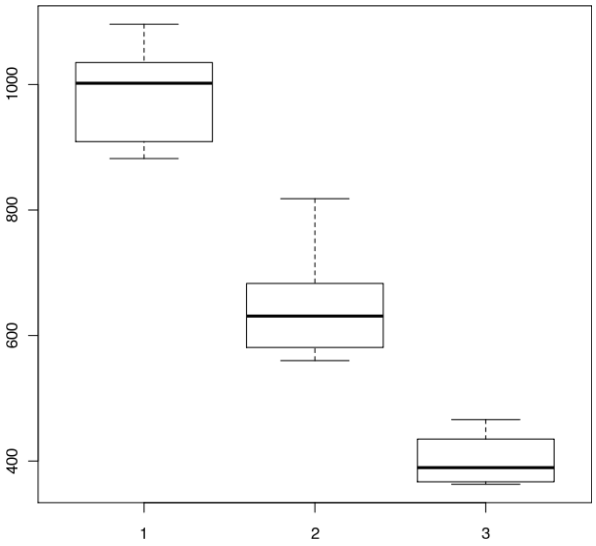
**Beneficios**



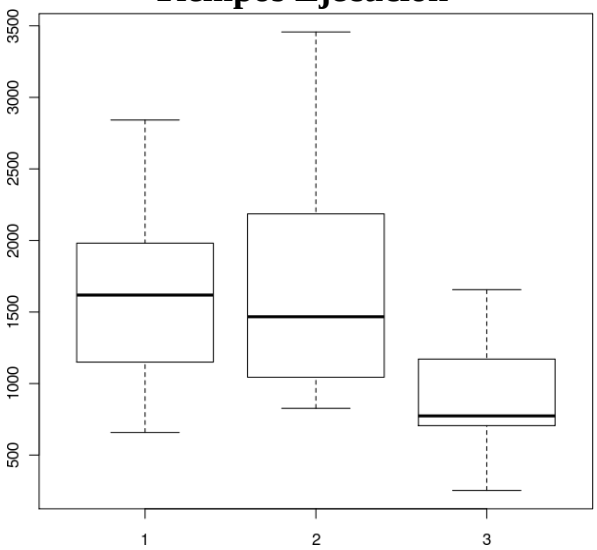

**Pedidos Asignados**




**Horas Absolutas**




**Tiempos Ejecución**



# 6.8. Influencia de la hora de las peticiones

En el problema podemos tener algunas horas que tengan más peticiones que otras. Dado el escenario del primer experimento, donde la flota de camiones y los pesos de las peticiones se distribuyen de forma equiprobable, y con una cantidad de peticiones de 250, generaremos variaciones en las que 4 de las horas tengan el doble de probabilidad de tener peticiones que el resto y compararemos las soluciones con las obtenidas en el primer experimento. Usaremos el heurístico de maximizar beneficios y el algoritmo de Hill Climbing.

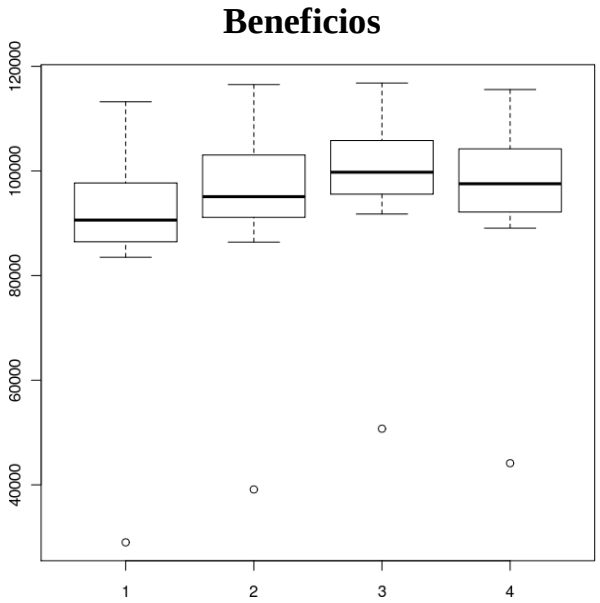
| | |
|---|---|
| **Observación** | Queremos observar como varían los resultados si existen 4 horas del día que contienen el doble de peticiones. |
| **Planteamiento** | Crearemos cuatro escenarios iguales pero con las cuatro horas con el doble de pedidos distribuidas de distinto modo. |
| **Hipótesis** | Con las distintas distribuciones se apreciarán distintos resultados. |
| **Método** | - Elegimos 10 semillas distintas, una para cada réplica.<br>- Elegimos 250 peticiones<br>- Elegimos los pesos de las peticiones de forma equiprobable.<br>- Elegimos los camiones de forma equiprobable.<br>- Elegimos la solución inicial Greedy 2.<br>- Usamos el algoritmo de Hill Climbing.<br>- Usamos el heurístico de maximizar ganancia.<br>- Experimentamos con 4 escenarios en función de las horas con pedidos dobles:<br>{ 8 , 9 , 10, 11 } { 11 , 12, 13 , 14 } { 14 , 15, 16 , 17 } y { 9 , 11 , 14, 16 }<br>- Ejecutamos 10 iteraciones de cada réplica con cada una de las 10 semillas.<br>- Hacemos el estudio de los resultados obtenidos (Beneficios, Horas absolutas, Tiempo de ejecución y número de pedidos asignados). |
| **Resultado** | Los mejores beneficios los ofrece la distribución en la que mas pedidos pueden entregarse mas tarde. En cambio, la diferencia de horas absolutas se reduce cuando los pedidos son a primeras horas. |
| **Conclusiones** | Aunque el número de pedidos asignados sea similar en todos los casos, si que se aprecia un aumento de los beneficios cuando las horas de entrega son más tardías, ya que ofrece más margen de maniobra.<br><br>Curiosamente sucede lo contrario con la diferencia de horas absolutas, donde se aprecia que cuando los pedidos son a horas más tempranas las horas absolutas son menores. Esto es debido a que nuestro heurístico y nuestro estado inicial procuran entregar el mayor número de pedidos a las primeras horas para maximizar el beneficio, pero no intentan entregar los pedidos a su hora. |

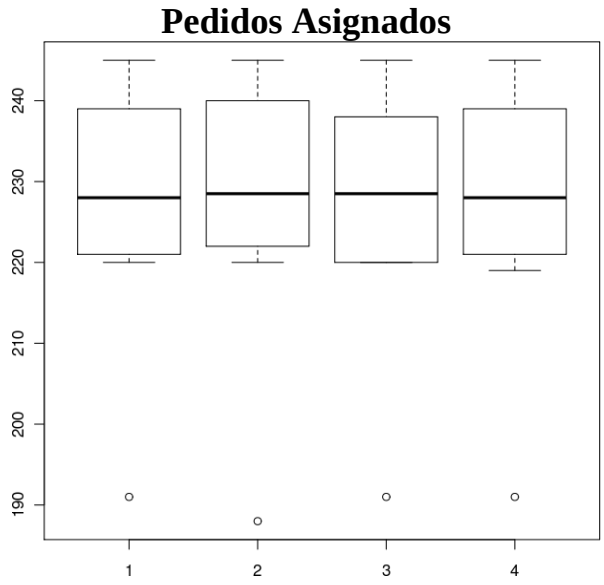
**Tablas utilizadas para el análisis:**

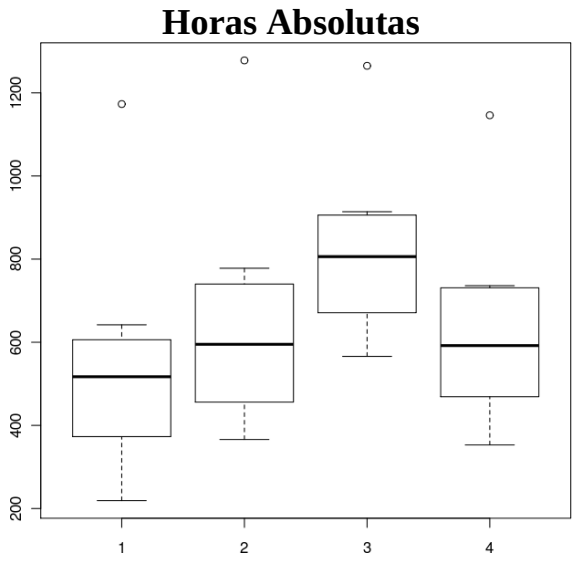
| Medias | Beneficios | Asign. | Horas | Tiempos (ms) | Pasos | Estados gener. |
|---|---|---|---|---|---|---|
| **8,9,10,11** | 88.012 | 227,2 | 529,1 | 1.712,1 | 3,2 | 43.033 |
| **11,12,13,14** | 92.957 | 227,2 | 648,2 | 1.764,1 | 2,8 | 38.696,8 |
| **14,15,16,17** | 97.034 | 226,4 | 812,6 | 1.214,6 | 1,6 | 25.727,7 |
| **9,11,14,16** | 94.628 | 227,1 | 622 | 1.269,8 | 2,1 | 31.449,8 |

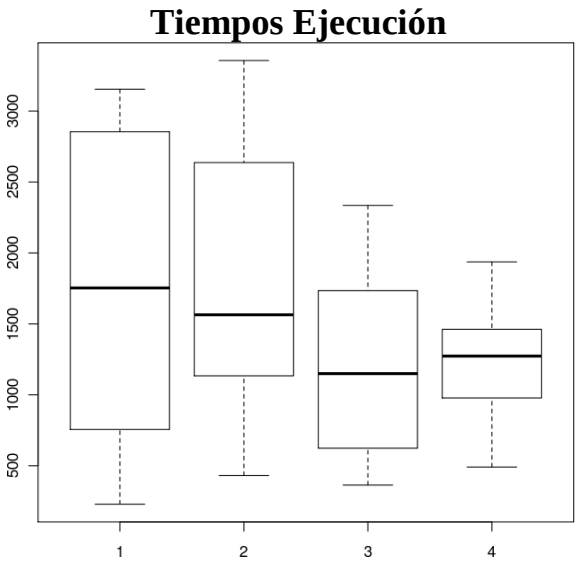
**Gráficos utilizados para el análisis:**

*Horas con el doble de probabilidad de contener pedidos:*
*1 - 08:00 09:00 10:00 11:00*
*2 - 11:00 12:00 13:00 14:00*
*3 - 14:00 15:00 16:00 17:00*
*4 - 09:00 11:00 14:00 16:00*



# 7. Conclusiones finales

Después de analizar los experimentos realizados se puede llegar a varias conclusiones, una de ellas es que el empezar con una casi-solución, como la que generan los greedy, hace que el proceso de análisis sea mucho más rápido y menos costoso. Empezar con un estado totalmente vacío puede hacer que genere estados intermedios que el greedy ya no puede contemplar y llevarnos por una solución alternativa, pero el coste que produce empezar un estado de cero es muy elevado.

Lograr un conjunto de operadores que recorra todo el espacio de soluciones consigue que el estado final que alcanza sea óptimo, por contra también provoca que el tiempo de proceso se dispare.

La función heurística debe contener todos los parámetros necesarios para que estado a estado nos vayamos acercando cada vez más a nuestro objetivo. Si no se controlan bien las ponderaciones de los parámetros usados se puede estar dando más valor a una solución potencial que a una real, provocando que no logremos nuestro objetivo.

El Simulated Annealing, pese a no probar todos los estados posibles, si se calibran bien los parámetros que necesita, se acerca a una solución muy similar a la que ofrece el Hill Cimbing. Ajustando estos parámetros podemos calibrar nuestra preferencia, si preferimos que la solución dada sea la mejor posible a costa de que la ejecución sea más lenta, o sacrificar parte de la solución para mejorar en velocidad.